# Heterogeneous exascale computing.

Ladislav Hluchý, Martin Bobák, Henning Müller, Mara Graziani, Jason Maassen, Hanno Spreeuw, Matti Heikkurinen, Jörg Pancake-Steeg, Stefan Spahr, Nils Otto vor dem Gentschen Felde, Maximilian Höb, Jan Schmidt, Adam S. Z. Belloum, Reginald Cushing, Piotr Nowakowski, Jan Meizner, Katarzyna Rycerz, Bartosz Wilk, Marian Bubak, Ondrej Habala, Martin Šeleng, Štefan Dlugolinský, Viet Tran, Giang Nguyen

Ladislav Hluchý, Martin Bobák, Ondrej Habala, Martin Šeleng, Štefan Dlugolinský, Viet Tran, Giang Nguyen
Institute of Informatics, Slovak Academy of Sciences
Dúbravská cesta 9, SK-845 07 Bratislava, Slovakia
e-mail: {ladislav.hluchy,martin.bobak,ondrej.habala,martin.selengstefan.dlugolinsky,viet.ui,giang.ui}@savba.sk

Henning Müller, Mara Graziani, Matti Heikkurinen
University of Applied Sciences Western Switzerland (HES-SO)
Sierre, Switzerland
e-mail: {henning.mueller,mara.graziani,matti.heikkurinen}@hevs.ch

Jason Maassen, Hanno Spreeuw
Netherlands eScience Center
Science Park 123, Amsterdam, Netherlands
e-mail: {j.maassen,h.spreeuw}@esciencecenter.nl

Jörg Pancake-Steeg, Stefan Spahr
Lufthansa Systems
Siemensdamm 62, D-13627 Berlin, Germany
e-mail: {joerg.pancake-steeg,stefan.spahr}@lhsystems.com

Nils Otto vor dem Gentschen Felde, Maximilian Höb, Jan Schmidt
Ludwig-Maximilians Universitaet
Munich, Germany e-mail: {felde,maximilian.hoeb,jan.schmidt}@nm.ifi.lmu.de

Adam S. Z. Belloum, Reginald Cushing
Informatics Institute, University of Amsterdam
P.O. Box 94323, 1090 GH, Amsterdam, Netherlands
e-mail: {a.s.z.belloum,r.s.cushing}@uva.nl

Piotr Nowakowski, Jan Meizner, Katarzyna Rycerz, Bartosz Wilk, Marian Bubak
ACC Cyfronet AGH, AGH University of Science and Technology
ul. Nawojki 11, 30-950 Krakow, Poland
e-mail: {p.nowakowski,j.meizner,b.wilk}@cyfronet.pl,{kzajac,bubak}@agh.edu.pl

**Abstract** Exascale services bring new unique challenges that the current computational, big data and workflow solutions are unable to meet. The chapter includes a detailed description of selected exascale services with known state of the art in extreme date solutions. The integration of requirements and the analysis of the state of the art in the exascale field is centered in on a description of a high-level architectural approach. The next main contribution of the paper is the description of the architecture capable to handle heterogeneous exascale services coming from both academic as well as industrial sphere. Those two models represent a (conceptual, and technological) design of a platform that addresses the requirements of the use cases. The resulting architecture will help us provide computing solutions to exascale challenges within the H2020 project PROCESS.

## 1 Introduction

The main aim of this chapter is to provide a description of services requiring heterogeneous exascale computing together with the architecture of an exascale system (it is the initial architecture of the PROCESS project[1]). Its design is driven heterogeneous exascale services coming from both academic as well as industrial sphere. The chapter starts with the analysis of our use cases as the main influencer of the architecture. This analysis is followed by the description of the initial architecture. It is depicted by two models represent a (conceptual, and technological) design of a system that addresses the requirements of the use cases.

The chapter has the following structure:

- Section 2 describes exascale learning on medical image data.
- Section 3 presents exascale challenges within the LOFAR data volumes.
- Section 4 discusses a supporting innovation based on global disaster risk data.
- Section 5 describes the ancillary pricing for airline revenue management.
- Section 6 presents the agricultural analysis based on Copernicus data.
- Section 7 concentrates on creating of the PROCESS architecture, with a high-level structure of the project platform.

## 2 Exascale learning on medical image data

Digital histopathology is the automatic analysis of a biopsy or surgical tissue specimens, which are captured by a high resolution scanner and stored as Whole Slide Images (WSIs). WSIs are usually stored in a multi-resolution pyramid structure, where each layer contains down-sampled versions of the original image. The amount of information in WSIs is huge, since it includes tissue that is not relevant for cancer

[1] PROCESS project homepage `https://www.process-project.eu/`

diagnosis (e.g. background, stroma, healthy tissue etc.) For this reason, machine learning and deep learning models are built to detect a Region of Interest (ROI) within WSIs. ROIs are portions in the WSIs where the cancer is spreading and therefore contain relevant information to train the network.

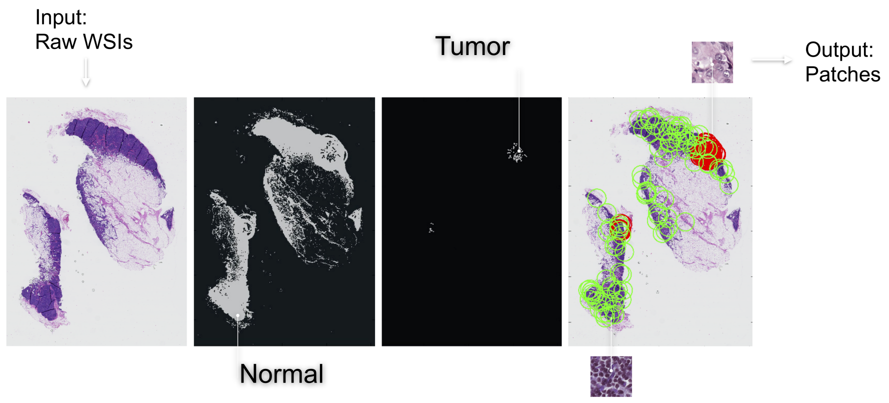

**Fig. 1** WSIs preprocessing pipeline: Normal tissue and tumor tissue masks are extracted and high-resolution patches are sampled from the selected regions.

Figure 1 shows an example of the data preprocessing pipeline. As a first step the raw WSIs are analysed at a very low resolution level, and tissue is filtered from the background. Based on doctorâĂŹs annotations, tumor regions are isolated. These regions represent ROIs which are used to perform network training. From the normal tissue and from tumor ROIs, patches are extracted at a higher level of magnification. Higher resolution highlights qualitative features of the nuclei which are essential for cancer detection. For instance, recent research has shown that the performance of classifiers improves wither resolution patches [12].

Exascale learning on medical image data aims at improving the performance of decision support in the process of cancer detection, localisation and staging/grading to optimize and personalize treatment planning. The automated tools deliver decision support for physicians, speed up visual inspection of tissue specimens and reduce subjectivity in the grading/staging process.

The use case tackles cancer detection and tissue classification using histopathology images, such as CAMELYON and TUPAC. The PROCESS infrastructure allows us to develop more complex models and to use increasingly larger amounts of data at a finer scale.

Access to exascale computing will consistently decrease the turn-around time of the experiments, pushing the boundaries of computation and consequently allowing researchers to develop models that are otherwise computationally unfeasible.

The main focus of the use case and technologies will be on the application of content-based tasks on WSIs, which are gigapixel images. As an example, three main tasks will be addressed:

- **Search** - e.g. “Are there patients with comparable scan results? Are there corresponding study cases?”
- **Detection** - e.g. “Where can cancerous tissue be detected? Are there cancerous changes visible in this scan?”
- **Classification** - e.g. “Which stage does this cancer belong to?”

A typical use case scenario can be defined as follows: âĂIJThe researcher wants to train a deep neural model from a large corpus of WSIs (»1TB). The training procedure requires intense prototyping for the customisation of the neural network architecture, learning criteria and dataset preprocessing. In addition, handling WSI datasets requires efficient storage of large datasets.âĂİ

To make an example, approximatively 1000 WSI are included in the CAMELYON17 dataset, 500 of which contain annotations and metadata (such as a description of the patient case, image width and height, the number of resolution levels and a text report with the main findings in the image). Tumors as small as 100x100 pixels should be detected in 200,000x100,000 gigapixel microscopy images. A brute-force approach would perform pixel-by-pixel analysis, requiring the classification of more than 109 elements. If the state-of-the-art model for image classification is used, e.g. ResNet [8], the number of operations needed for training is on the order of 1017, with at least 3.6 GFLOPS required per each full scan of the training data (i.e. epoch), which may increase to 11.6 GFLOPS if the depth of the network is increased. Hence, with 50 training epochs, the computations required to scale up to the order 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 [19]. Hence the requirements for intensive computations and massive storage.

## 2.1 Medical use case motivation

The motivation is to improve the performance of automated cancer diagnostics and treatment planning. We plan to develop more powerful tools for cancer detection, localisation and stage classification, which could support the decisions of physicians during the diagnostic process. Cancer diagnostics are generally time-consuming and suffer from high disagreement rates between pathologists, since tumour stages do not have defined boundaries [9]. Through 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 resources brought by the connection between High Performance Computing (HPC) and medical imaging research would permit analysis of large-scale challenge datasets for medical imaging. The training of multiple deep learning models could be distributed among different computational centres and more complex models could be trained with shorter time constraints.

## 2.2 Goal of the medical use case

The final goal is to improve the performance of the current state-of-the-art algorithms for cancer detection through the accomplishment of three main objectives. The first objective is the development of image analysis algorithms for supervised and weakly supervised learning. Supervised learning attempts to learn a mapping f(.) between each datapoint x (e.g. the histopathology image) and its label y (e.g. presence of cancer, cancer stage, cancer locations) such that y = f(x). The introduction of weak labeling will enable the use of scarcely annotated data to improve model performance and generalisation. To further build upon these advantages, unsupervised learning algorithms can be used to model unannotated data, thus removing the need of manual annotation before training. The trained networks are used on testing data to identify tumorous regions and to suggest pathologists possible tumorous regions in the histopathology image. Visualisation methods can be used to gather instantaneous feedback and explanations for the network decisions, thus increasing the reliability of the models and improving trust in the developed tools [13]. The second objective is the distribution of training across multiple computational centres. Neural networks can be independently trained at different computational centres, obtaining a speed up in time proportional to their number. The models could then be assembled into a single model with improved generalisation abilities. The third objective consists of investigating the trade-off between performance and computational resources by increasing the size of the datasets and the computational complexity of the models. For example, an enormous amount of computations is required if slight perturbations are introduced in the data and multiple models are trained to investigate model robustness. Overall, the presented objectives will contribute to the analysis of huge collections of medical data, and therefore to the development of improved algorithms for the automatic suggestion of patient cancer stage and correlated patient cases.

## 2.3 Model training workflow of the medical use case

Different Deep Neural Networks (referred to as “networks” in this section) need to be trained and monitored. For this workflow, we identify three main phases and three different application scenarios. The three phases consist of:

- **Phase 1: Patch Extraction** - thousands of high resolution patches are extracted from raw WSIs
- **Phase 2: Local and Distributed Training** - supervised and unsupervised training is performed on the extracted dataset of patches in both local and distributed fashion, and training statistics are monitored

- **Phase 3: Performance Boosting** - multiple training runs are relaunched with data perturbations to address model robustness

The three different scenarios for the workflow can be pictured as follows:

- **Scenario n.1**: Moving the models to hospitals. Different models are trained independently at each data centre and finally ensambled and made available to hospitals for local testing.
- **Scenario n.2**: Distributing training to the data location. Training is distributed across centres on different portions of data and computations are dynamically handled. This scenario simulates the distribution of training to where the data resides, especially in cases of private data that cannot leave the hospital environment.
- **Scenario n.3**: Dataset sizes and model complexities are jointly increased over multiple training runs and pareto frontiers are investigated.

Figure 2 summarises the DNN training workflow. Network development is currently performed locally on less powerful infrastructures, using a series of tools for data processing and manipulation (reported in Table 1). The libraries that are generally involved in network training set the requirements for a user-friendly infrastructure where it is possible to create virtual environments that can replicate the development environment. Docker containers can be used to create separable, self-contained virtual environments in which to develop software and run programs. The 'image' of a Docker module can contain standalone and ready-to-be-executed packages of software, tools, libraries, and settings. The flexibility of these containers allows porting complex software development environments required by Machine Learning and Deep learning.

Moreover, network training calls for a service-oriented infrastructure that is easily scalable to large computations. The setup of a Hadoop ecosystem will scale up to larger datasets within the HPC and Cloud computing infrastructures.

**Table 1** Overview of current technologies and local existing infrastructure.

| | |
|---|---|
| **Currently used technologies:** | Python 2.7, Tensorflow 1.4.0, Caffe, Theano, Lasagne, DIGITS, mxnet, Keras 2.1.2, TFLearn, Numpy, SciPy, Pandas, Scikit-learn,OpenCV, Openslide, Matplotlib, Seaborn, Skimage, h5py, PyTorch, OpenBLAS, cuDNN, FFmpeg, NLTK, Gensim, R, Weka. |
| **Data Storage:** | NAS. |
| **Data Processing:** | H5DS, Apache Spark, Hadoop. |
| **Existing computing infrastructure:** | 8 GPUs. |

Data preprocessing is the second step in the workflow and belongs to the Patch Extraction phase. In this phase, many of the technologies mentioned in Table 1 are used to process the information in the WSI. The intermediate results of image preprocessing (ex. patch extraction, data augmentation output, image metadata) need to be stored so that they could be used for multiple training runs. Data preprocessing is intrinsically parallel and should be scaled to multiple CPU cores.

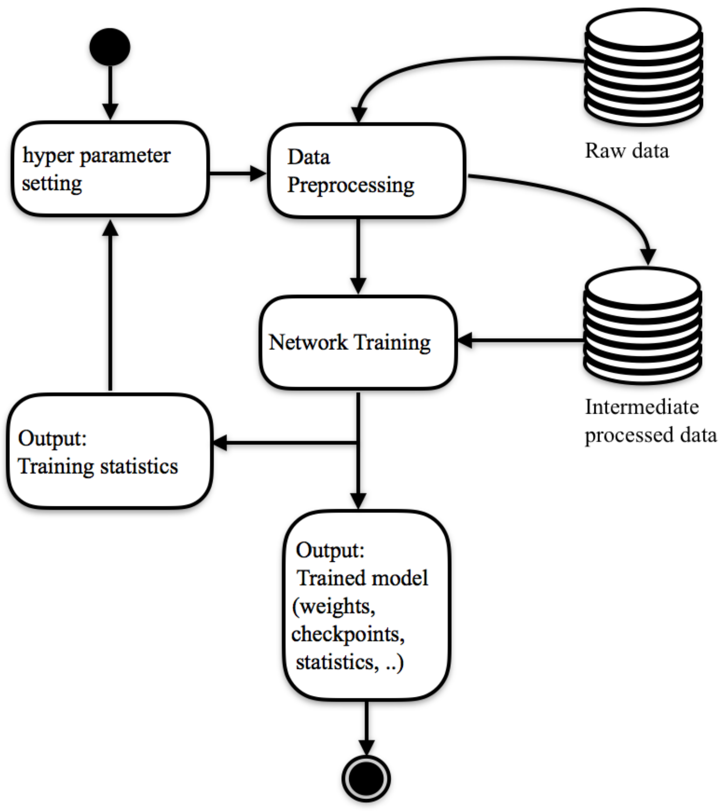
hyper parameter setting
Data Preprocessing
Raw data
Network Training
Intermediate processed data
Output: Training statistics
Output: Trained model (weights, checkpoints, statistics, ..)

**Fig. 2** Network Training workflow: The solid dot represents the starting point in the workflow and the dot surrounded by a circle represents the ending point.

The third step in the workflow is network training, which requires as input a configuration file containing a list of model settings and the WSI data (both metadata and images). The network training process itself is generally highly parallelizable. The parallelization might be performed at different levels and could involve both CPUs and GPUs. For instance, the simplest training configuration settings should enable parallel training on a single node with multiple GPU cores. More sophisticated parallelization methods would scale the computations on different nodes. Parallelization could then be performed at different scales, namely at the data level and at the model level. On the one hand, parallelization at the data level involves the use of different nodes to run the same portion of code on different batches of data. In this context data storage may represent a huge bottleneck in the organization of the clusters, and therefore the PROCESS Data Manager should be able to identify and tackle possible issues. On the other hand, the parallelization could be performed at the model level, involving the development of more sophisticated models that distribute the tasks among different workers. However, this method is generally discouraged since the communication latency between cluster nodes might be a limiting factor. Therefore, the development of models that could efficiently use the computational power of the PROCESS infrastructure stands as a perfect use case for the project, involving a series of challenges in the development that still need to be tackled. The training will also be distributed across the multiple centres. As a first scenario (Scenario n.1), different models will be trained independently on each data centre and the final results will then be ensambled in one single model with higher generalisation. This scenario is close to the hospital reality where the hospitals could download the trained models and deploy them locally without the need to share the data. In Scenario n.2, training is distributed across the centres and the computations are dynamically handled. This scenario will highlight the challenges of distributing computations on physically distant centres. Dedicated handling of job queuing and data routing is needed to organise the computations among nodes. The final scenario, (Scenario n.3) addresses the task of constantly increasing model complexities and dataset sizes. For instance, the number of extracted patches will increase and data augmentation will be used to test model performances for increasing dataset sizes. Moreover, the different resolution layers will be used to create more complex models and performances will be recorded for a growing number of model parameters.

In all the scenarios, network training will be monitored in real time, therefore a service to visualise statistics and collect feedback is needed.

The final output of training consists of the trained network model (e.g. weights, checkpoints, training statistics), which should be stored in the system for testing. The trained model should be made available for download for local testing.

The convention for the model saving format generally includes a series of HDF5 files on the order of several MB per file, often scaling up to GB when larger models are trained. The use case should not exclude the possibility to train several different models and store the results in a dedicated file system.

The different stages in the pipeline and the three application scenarios are summarized in Figure 3.

## 3 Square Kilometre Array/LOFAR

Analysing the massive volumes of data stored in the archive is an acute problem. For example, even with the LOFAR data volumes (currently 28 PB), a significant percentage of the archived data is never used, mostly because only expert users are capable of accessing and processing such data.

The LOFAR radio telescope consists of around 8000 antennas in 51 stations. These antennas produce approximately 25 GB/s, which needs to be processed in real time to combine their signals into a single view of the sky. This data is stored in the

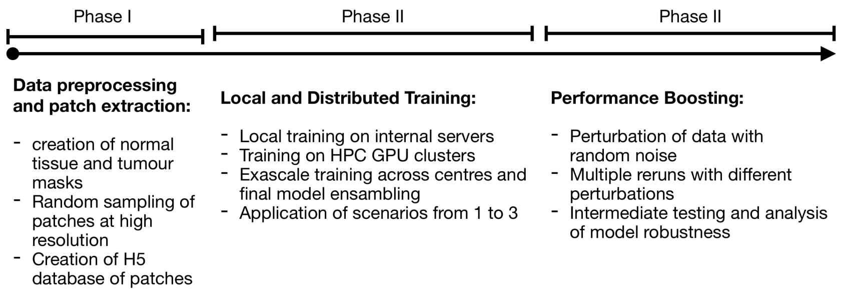
Phase I
Phase II
Phase II
Data preprocessing and patch extraction:
- creation of normal tissue and tumour masks
- Random sampling of patches at high resolution
- Creation of H5 database of patches
Local and Distributed Training:
- Local training on internal servers
- Training on HPC GPU clusters
- Exascale training across centres and final model ensambling
- Application of scenarios from 1 to 3
Performance Boosting:
- Perturbation of data with random noise
- Multiple reruns with different perturbations
- Intermediate testing and analysis of model robustness

**Fig. 3** Workflow Phases and Application Scenarios.

LOFAR Long term archive (LTA), which is distributed over Amsterdam, Juelich and Groningen. Data is typically stored on tape and can be accessed via a web portal or SRM client. The data remains private to its originating project for a year following acquisition but is made public afterwards.

Individual observations produce up to 100TB of data. These are typically reduced to 16 TB before being stored in the archive. Each observation is stored in a well-defined directory structure, containing multiple files with both the metadata and the measurements themselves.

A major hurdle is that the archive stores partially processed observations (also referred to as visibilities), not the images or sky maps that astronomers typically use as a starting point for their research. Once unknown sources are detected in the sky maps, the astronomers often need to run more detailed analysis on the visibilities directly. For this reason, the visibilities are stored in the archive, not the sky maps.

The initial conversion of the observations into sky maps requires several processing steps, such as retrieving the data from tape, performing RFI (Radio Frequency Interference) removal, calibration, and imaging. Unfortunately, due to the large data volumes (tens to hundreds of TBs), and complexity of the tools, each of these processing steps take a significant amount time and effort by the astronomer. Further automating this part will significantly simplify the use of the stored observations, and allow the astronomer to focus on the science instead of the preprocessing of the data.

## 3.1 SKA use case motivation

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. The LOFAR LTA consists of tapes at locations in Amsterdam (The Netherlands), Jülich (Germany) and Poznan (Poland). Its current volume is about 28 PB. This consists mostly of “Measurement Sets”, i.e. visibilities - correlated signals from LOFAR stations. LOFAR is one of the testbeds for the Square Kilometer Array (SKA) and is similar to the part of SKA that will be built in Australia.

Initially, the data produced by LOFAR is reduced and analysed by the astronomers who requested the observations. Subsequently, the data is ingested in the LTA. Since LOFAR is a wide field radio telescope, the LTA should contain a wealth of serendipitous discoveries, like transient radio sources. Such sources can only be discovered by massive analysis of data from these observations.

The first step in the analysis is often the creation of images from the data which the astronomer can inspect. Unfortunately, significant processing and expert knowledge is needed to produce these image cubes from observations stored in the archive. As a result of these difficulties, the LOFAR LTA is largely unexplored by astronomical research. We want to unlock the LOFAR LTA and increase its scientific output.

## 3.2 Goal of the SKA use case

The goal of this use case is to simplify the processing of archived data. Astronomers should be able to select a dataset on a portal, select a workflow, and then launch the processing pipeline from there. For this, we need an easy to use, flexible, efficient and scalable workflow infrastructure for processing of extremely large volumes of astronomical observation data.

PROCESS will provide a mechanism to run containerized workflows, thereby improving the portability and easy of use. A suitable portal is needed to select datasets and workflows. Through this portal, the astronomer must be able to browse through the available datasets and available workflows, and launch processing directly from there to the hardware infrastructure available in the project. Data should then be transferred from the LTA to the processing infrastructure, processed, and the results made available in the portal.

Currently, the processing of a dataset typically takes much longer than the observation time used to acquire the dataset. For example, to process a single 8-hour observation using the calibration and imaging workflow described below currently takes about 4 days (due to the lack of parallelisation of part of the workflow). To keep up with the speed at which data is generated, it is essential to increase the processing performance. Running the same workflow in parallel on different dataset provides the horizontal scalability required for processing the LOFAR archive, and (in the future) the SKA archive as well. Vertical scalability will be achieved by applying multi- and many-core techniques.

## 3.3 SKA use case scenario

In a typical use case an event is detected in the sky, for example a satellite detects a flash of gamma radiation at a certain position. Subsequently, astronomers want to observe this same patch of the sky with other instruments, such as optical and radio telescopes. Additionally, the astronomers want to check past observations to determine if any objects had previously been detected at the given position. To do so, data must be retrieved from the archive that covers this position on the sky, and be converted into sky maps.

Besides inspecting single observations, astronomers also perform surveys where a large number of observations are combined to analyse certain features or detect certain events in large portions of the sky. For such surveys, a large number of observations must be processed by the same workflow, for example to create larger sky maps.

### 3.3.1 The workflow of the SKA use case

The current pipeline is developed by Leiden University and SURFsara in the Netherlands. It is based on the following software:

- uberftp client [2]
- globus-url-copy client [3]
- voms-client (only on the login node) [4]
- CVMFS [5]
- PiCaS [6]
- python 2.7 or higher
- pre-FACTOR [7]
- DDF [8]

A gridFTP enabled client is required for the interaction with the Grid storage (dCache). Voms tools should be installed and configured to support the LOFAR VO to allow users (or 'robot users') to create proxies with LOFAR attributes. Installing and configuring CVMFS with Softdrive mount point enables access to the LOFAR software tree from any compute node.

[2] http://www.lofar.org/wiki/doku.php?id=public:grid_srm_software_installation

[3] http://www.lofar.org/wiki/doku.php?id=public:grid_srm_software_installation

[4] http://www.lofar.org/wiki/doku.php?id=public:grid_srm_software_installation

[5] https://cernvm.cern.ch/portal/filesystem

[6] http://docs.surfsaralabs.nl/projects/grid/en/latest/Pages/Practices/picas/picas_overview.html

[7] https://github.com/lofar-astron/prefactor

[8] https://github.com/mhardcastle/ddf-pipeline

A single server runs the PiCaS system which uses the CouchDB database for storing the descriptions of jobs. Python is required to execute the staging scripts and interact with the CouchDB API via a python-based client. The processing clients require outbound internet connectivity to retrieve job descriptions from PiCaS.

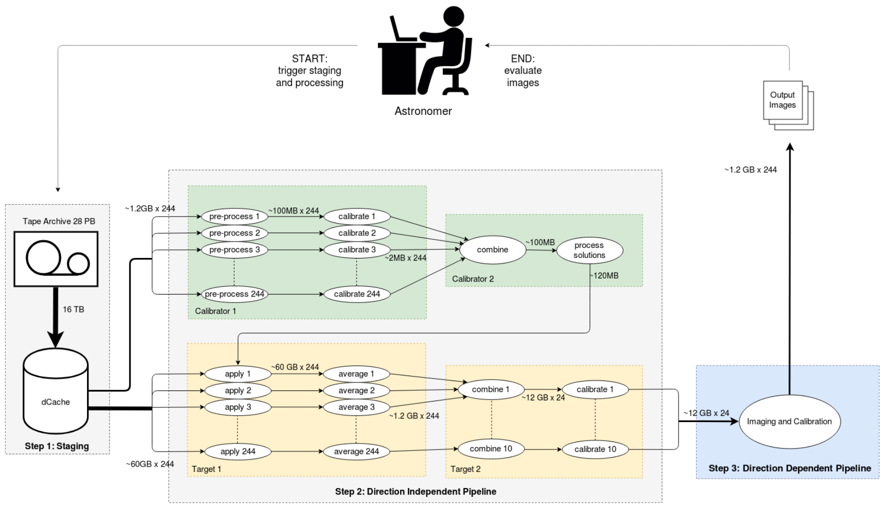

**Fig. 4** The calibration and imaging pipeline for a single observation: The astronomer triggers the staging of a 16 TB observation from the archive to temporary storage. This data is then first processed by the Direction Independent Pipeline which may run in parallel on up to 244 nodes (taking approx. 4 hours). Next, the Direction Dependent Pipeline is run on a single node (taking approx 4 days) and produces up to output 244 images of 25K x 25K resolution for inspection by the astronomer.

As a first step in the processing pipeline (shown in Figure 4), the observation data needs to be downloaded from the archive. This may require retrieving the data from tape and storing it on temporary storage (dCache). From there, the data is accessible via GridFTP.

The workflow consists of 2 pipelines, the Direction Independent Pipeline (DI), and the Direction Dependent Pipeline (DD). Both pipelines perform calibration, which is essential to detect the weakest possible signals. Calibration is needed to remove interference from the instrument and the ionosphere, thereby increasing the sensitivity of the telescope significantly.

Calibration starts by converting a sky-map of well known radio sources into visibilities, using a Fourier transform to convert to (u, v, w) space, and interpolating from a regular grid to the irregular grid measured by LOFAR. This gives the “ground truth” visibilities. Next, the observed visibilities of the well-known sources need to be mapped to match this ground truth. The mapping matrix is derived through a least-squares fitting algorithm (such as Levenberg-Marquardt). The DI pipeline creates a single mapping matrix which is applied to the entire observation. This calibration serves as a starting point for the DD pipeline, which performs a similar calibration for dozens to hundreds of directions within the LOFAR array beam.

The outputs of both pipelines are shown in Figure 5 below. Although images may be produced after DI, they are limited in resolution and contain residual errors caused by disturbances in the ionosphere which vary over the field of view. This is shown the left image in Figure 5. The DD takes the output of the DI as input and further reduces the errors to produce high resolution, low error images, as shown in the right image of Figure 5.

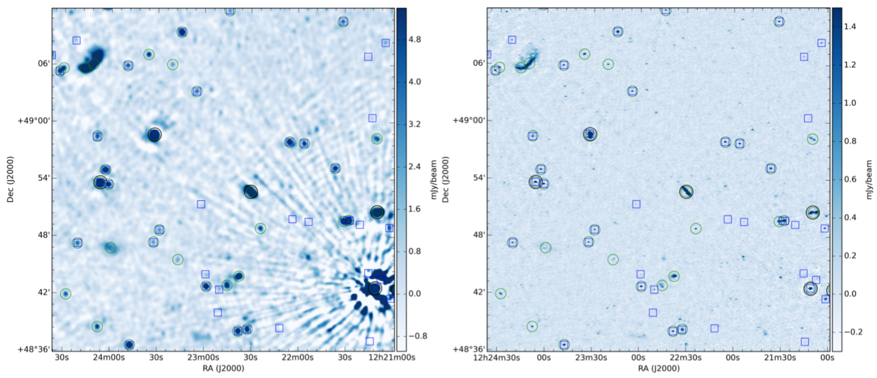

**Fig. 5** Output images produced after Direction Independent Calibration (left) and additional Direction Dependent Calibration (right). The image left clearly contains artefacts caused by the ionosphere. These are subsequently removed by the direction dependent calibration, shown on the right. Image courtesy of [15].

DI uses the pre-FACTOR package to perform the initial calibration. Internally it consists of several steps, some of which can be run in parallel. As shown in Figure 4, it splits the input data into 244 subbands which can be processed independently. Each subband becomes a separate calibration job. Each job requires a relatively small input of 1.2 GB, which is reduced to an output of around 3 MB. When all jobs are finished, a second calibration job combines the results into a single table approximately 100 MB in size.

This table is then applied to the observation target, which again is split into 244 jobs, one for each subband. Each job requires around 60GB of input, and produces around 1.2 GB of output. The output of these jobs is then again combined into the final DI calibrated datasets, which consist of around 300 GB of data. The entire DI takes around 4 hours for a single dataset when running on 244 nodes.

Following the crude direction independent (DI) calibration, direction dependent calibration (DD) is performed. This is an iterative procedure on the target field, which is called self calibration. It is currently implemented using the DDF package on a single node (using all the cores of one high-end CPU). This takes about four days uses the initial calibration output (300GB) to reduce the 16 TB of observation data to 0.5 TB. This data is then converted to a collection of (up to) 244 images at 25k x 25k resolution (1.2 GB each).

Currently, the processing is performed on the Gina cluster of SURFsara, which is located in Amsterdam. Where the jobs of the DI pipeline can be run on relatively simple compute nodes with 8 GB of RAM and 100 GB of scratch space, the DD pipeline requires a single high-end machine with at least 256GB of RAM and 3 TB of scratch.

To process a single observation, approximately 27K core hours are required to run the pipeline. To process the complete archive an estimated 47M core hours is currently needed. An additional 8 to 12M core hours is needed each year to keep up with the 5-7 PB of data produced yearly. Note that these estimates are for running a single pipeline on all data, using a single configuration. However, for different science cases different configurations or pipelines are used. Therefore, this estimate is a lower bound for the processing time that is required.

## 4 Supporting innovation based on global disaster risk data

There is, at the moment, considerable uncertainty related to mechanisms the GAR-like data is produced and consumed as part of the global risk assessment process. The amount of data is conceptually based on the following formula:

*"Relevant surface area of area under analysis "X" resolution "X" number of scenarios needed for probabilistic analysis "X" size of an individual scenario data"*

The number of scenarios and the necessary resolution of the scenario data depends on the type of hazard. For example, in case of analysing the earthquake risk it is not necessary to pinpoint the exact position of the epicentre, as differences of the order of few kilometres do not influence the possible outcomes in case of a major earthquake. However, the impact of a flash flood scenario may be dramatically different depending on small variations of the location of the maximum rainfall, surface structure and e.g. slight variations in the elevation of building and infrastructure will have a major impact on the outcomes. Thus a global flood model will always unavoidably be a compromise (some of these compromises are discussed in detail by [14, 7]. The GAR 2015 dataset size ( 1.5TB) should be seen as the minimum, with the follow-up activities likely increasing the storage requirements by at least an order of magnitude during the duration of the PROCESS project.

The PROCESS project supports the distribution of the GAR 2015 datasets using a simple web portal. However, the plans for GAR 2019 will present additional challenges in addition to the likely increase of resolution of analysis:

- Loss calculation process will become a community-based effort aiming at scientific consensus (along the lines of the IPCC approach). Thus, there will be several datasets, produced through different methodologies and, at least initially, residing in different repositories (instead of a centrally managed system curated by UNISDR)

- The interaction between different research groups will almost certainly lead to the need for supporting versioning of the datasets as the process of cross-correlation and analysis uncovers opportunities for refinements and improvements in the accuracy of the models. However, for traceability reasons all versions of the data would need to be Findable, Accessible, Interoperable and Reusable (so-called FAIR principle, promoted by the RDA).

Thus, the key to the success is to showcase lifecycle support for disaster risk management data. This showcase would be aimed at convincing the data producers to apply solutions that are interoperable with the PROCESS approach (or, in an ideal case, adapt the PROCESS solution or PROCESS service). The initial key focus area will be in supporting the risk modelling community with the analysis of external reuse, based on the data published using PROCESS tools becoming more active towards the end of the project.

The UNISDR collaboration aims at solving two interrelated problems: how to support community uptake and broad range of innovation activities based on the new natural disaster risk related open data products, and how to enable more efficient data management of the disaster risk data in the face of increased amounts of data that will be produced in a more dynamic fashion by a distributed, heterogeneous collaboration.

## 4.1 Goal of the UNISDR use case

The use case goals are a) to increase the efficiency of data generation and curation (simulated data related to natural hazards) for probabilistic analysis purposes by the parties affiliated with the UNISDR Global Assessment process and b) to encourage and facilitate the third-party use of these datasets.

## 4.2 Workflow of the UNISDR use case

The pre-2017 workflow of the overall GAR process is presented in Figure 6. PROCESS support focuses on interaction and data sharing between the distributed hazard modelling teams and the risk computation team at UNISDR. Once the GAR process has been finalised, the hazard, exposure and vulnerability data would be turned into open data products.

The post-2017 GAR workflow differs considerably from this model (see Figure 7). The consensus process will encompass numerous teams performing analysis on a wide variety of datasets (many-to-many relationship between datasets, teams, and analyses). The UNISDR/CIMA process represents just one of these combinations, albeit due to its scale and nature as a recognised UNISDR project, it is a more

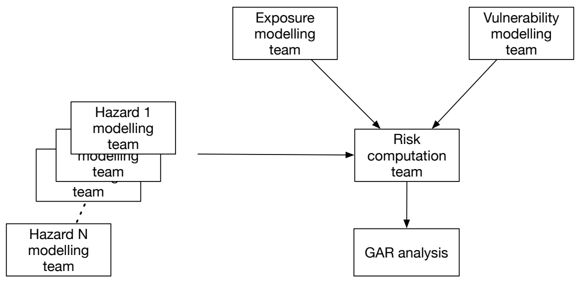

**Fig. 6** The pre-2017 GAR workflow.

far-reaching effort than most of the contributions to the 2019 version of GAR. From the PROCESS point of view, the successful completion of the data generation is important mainly as a way to establish the relevance of the project in the broader GAR community and to build foundations for deeper collaboration.

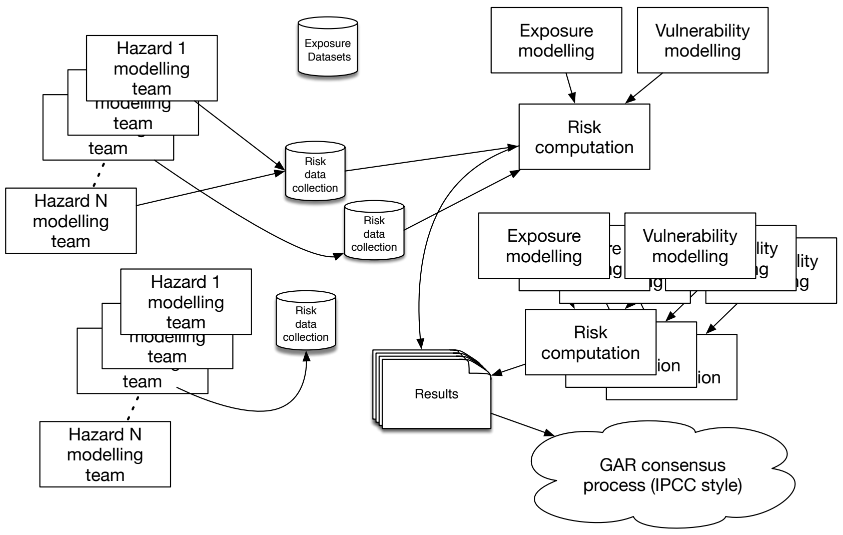

**Fig. 7** The post-2017 GAR workflow.

## 5 Ancillary pricing for airline revenue management

Ancillaries is a broad term for any services that goes beyond simple transportation from A to B. Ancillary in this sense can be anything from being able to check-in an additional bag to booking an "Uber" that transports the customer from the airport to his hotel. Although ancillaries have caught the attention of basically every airline, traditional RM software does not cover this topic at all, yet.

The challenges we face are manifold. We need to store and analyse huge data volumes that major airlines already have today (e.g. Deutsche Lufthansa has more than 100 million passengers per year). Due to the advanced digitization, we expect an exponential increase of the data volume. Beside the more dynamic or volatile booking data (we call this a passenger name record PNR) the solution needs a wide access to more data like airport basic data, vacation period in different countries, events like fairs, huge sport events etc. Also data like weather forecasts (e.g. snow conditions in the Alps) can be of interest. Also input from highly volatile social media streams like Twitter can be an interesting source to be analysed. And last but not least web-scraping can provide information about the airline's competitors and their offerings.

Airlines are not operating alone. Major airlines cooperate with other, bigger and smaller, airlines in joint ventures, networks and strategic alliances (e.g. the Star Alliance, One World etc.). Therefore, the data created in and by such networks needs to be included in the analysis and dynamic pricing solution.

To achieve this, we need an infrastructure that is capable of collecting, processing and analysing huge amounts of data. This needs to be done in a very dynamic way because the result of the analysis (i.e. the calculated price of an ancillary) must be feed back into the shopping / booking process. Due to the growth of on-line retailers and an increasing number of on-line travel platforms, airlines expect a yearly increase in the quantity of requests by about 20%. The infrastructure must be capable of responding to more than 100 million requests per day and within 500 ms per request.

Within the dynamic pricing engine we must make use of new prediction methods that offer ancillaries to a customer at an optimal price depending on his/her shopping history and individual preferences, and by also considering attributes of the requested itineraries, e.g.

- free golf bag for famous golfing destinations,
- or, free skiing bag if the passenger travels to e.g. Innsbruck and there is snow in the Alps.

We need to develop new algorithms and forecasting models to optimally price the provided ancillaries. This will include leading-edge methods and technologies from the domains of artificial intelligence and machine learning - like deep learning, decision trees, support vector machines, neural networks etc.

To evaluate these algorithms and models we need training and test data with a well known structure. Therefore the first step will be the development of a data generator for airline ancillary data. This is also necessary due to the high data protection level for customer-related data within the EU.

Finally, we will develop methods that assign a "booking probability" to each ancillary, since the customer only wants to be presented with ancillaries that match his/her preferences. It is obvious that if the customer is annoyed by the offered products (for example there may be too many of them, or they may not interest the customer), the likelihood of purchasing an interesting ancillary goes down significantly.

## 5.1 Ancillary pricing use case motivation

In traditional indirect distribution, airlines publish some of their data via third party systems, alongside their own internal source of availability. It is then up to other third parties, Global Distribution Systems (GDSs), to use these sources of data to create itineraries and to correctly apply fares to them. It is only at the point that the customer makes a booking that the airline has visibility on that customer and their request. At this stage, the customer decisions have been made based on the airlineâĂŹs static data it published in advance and aggregated without the influence of the Airline. The data is static with respect to content (combination of services offered as a bundle) and price (pre-determined in advance and not at the time when the shopping request is processed).

For this reason, airline revenue management (RM) has been mainly around the following aspect for more than 40 years: determining how much of seat inventory shall be sold for each of the previously determined static price-points. During the last decade - and mainly driven by on one hand the growth and success of so-called low-cost airlines, and on the other hand the increasing volume of flights booked on the airline's own website (were the airline has more or less full control about content and look and feel) - the focus has shifted from controlling how much of flight inventory to sell at pre-determined price-points to:

- merchandising any service item that might be related to a passengers journey, (e.g. rental car, valet parking, taxi from/to airport, lounge access, travel insurances etc.),
- unbundling of services, i.e. offering more choice options to the traveller by offering a large number of "à la carte" options (some of them could even be sold without the need to book a flight),
- controlling the entire offer that is made to a shopper, i.e. content (services and conditions), form and price points (in case it is not a single price but multiple prices because of e.g. "à la carte" options and ancillaries).

This means that selling ancillaries around the flight has become more and more important. For many airlines selling ancillaries has already become the main contributor to their contribution margin.

### 5.2 Goal of the ancillary pricing use case

The first goal of the use case is an analysis of current ancillary sales and hidden sales pattern. Therefore two approaches will be pursuit: first we evaluate the option to generate artificial sales data based on internal airline knowledge and secondly we try to evaluate real sales data from partner airlines.

The second goal is deriving a promising machine learning algorithm for pricing of offered ancillaries. Therefore we will train common machine learning algorithms (e.g. random forest and neural networks) and evaluate their performance.

Lastly, a simulation should indicate the quality and chances of the pricing algorithms. Ultimately good quality pricing algorithms promise airlines significantly higher revenues from their ancillary sales.

All of these business goals should be achieved while providing 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.

### 5.3 Scenario of the ancillary pricing use case

The key software layers (see Figure 8) can be divided into (a) a batch layer, (b) a stream layer and (c) a service layer. While the batch layer is used to build up the model by evaluating huge historical data piles (e.g. stored in HDFS) the stream layer is used to receive a continuous stream of live data to refine the model and integrate it into the model store (located in the batch layer). The service layer is used as the interface between e.g. the booking engine and the ancillary pricing engine. For this the Ancillary Price Calculator accesses the model store to calculate the price.

Therefore these layers need to have capabilities for:

- the collection of dynamic and static data from different sources, like structured flat files, shopping behaviour (offers and corresponding orders), click-stream data, competitor information (e.g. from web scraping, fare filing), social media streams etc.
- pattern recognition, modelling, predictive analytic methods, and other AI methods
- the process to calculate the relation/dependency between contents of offers (services), offered prices and booking probability
- providing an interface to give access on ancillary price recommendations to booking engines and travel portals in real time

Most software and hardware infrastructures at airlines' revenue management departments are still based on concepts which emerged in the 90s and early 2000s. Usually, there is a big monolithic piece of software that is really hard to maintain and difficult to extend. Furthermore, data storage is organized in big Oracle instances

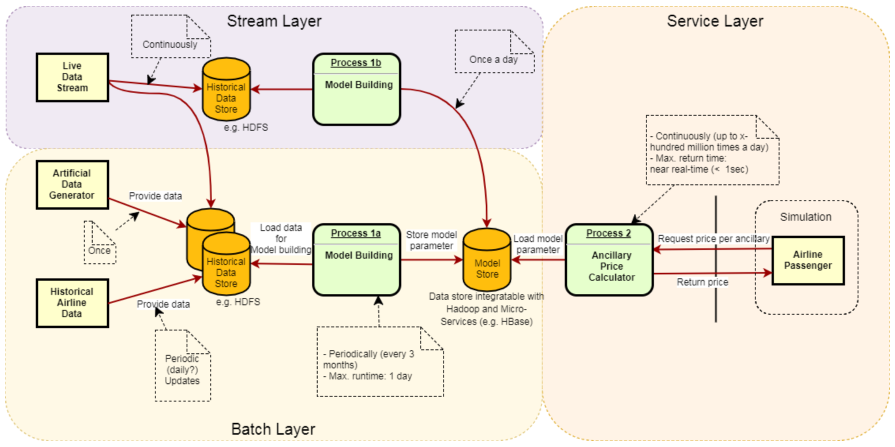
Stream Layer
Service Layer
Batch Layer
Continuously
Live Data Stream
Historical Data Store
e.g. HDFS
Process 1b
Model Building
Once a day
Artificial Data Generator
Provide data
Once
Historical Airline Data
Provide data
Periodic (daily?) Updates
Load data for Model building
Process 1a
Model Building
Store model parameter
Model Store
Data store integratable with Hadoop and Micro-Services (e.g. HBase)
- Periodically (every 3 months)
- Max. runtime: 1 day
Load model parameter
Process 2
Ancillary Price Calculator
- Continuously (up to x-hundred million times a day)
- Max. return time: near real-time (< 1sec)
Request price per ancillary
Return price
Simulation
Airline Passenger

**Fig. 8** Workflow scenario.

which guarantee data consistency and transaction safety, but at the drawback of poor scalability. The rapid growth of passenger numbers and the opening of new business fields has led to the following requirements for new software and hardware which traditional solutions cannot answer:

- Support for quick changes in the software: the market changes quickly and the internet allows the customer to easily compare offers. Hence, airlines have to react quickly to the market.
- Elastic scaling of hardware: growing hardware costs are no longer expected, since the machines are often left unused (e.g. few bookings with European airlines during night hours in Europe).
- Vertical scaling of data stores: the growing data volumes are approaching the limits of horizontal scaling of Oracle systems. Hence the need for vertical scaling.

The service layer (right-hand side of the workflow diagram) uses state-of-the-art microservice software stacks such as Spring boot, Kubernetes etc. and provides RESTful services. The service registry uses the Netflix Eureka framework and is replicated to avoid a single point of failure. Eureka also facilitates dynamic load balancing. The services are accessed through an application gateway, for this Netflix Zuul is used which also integrates security services.

The streaming and batch-layer is currently under development and therefore not yet fixed. It will likely rely on Hadoop/HDFS infrastructure and we plan to use HBase as an interface between the big data side and the microservice side. For the streaming layer, Apache Spark streaming looks promising. For the model building part, frameworks such as Tensorflow and H2o.ai will be evaluated.

## 6 Agricultural analysis based on Copernicus data

The fundamental scientific problem is the sustainability of food production on the global scale, which is a multi-disciplinary problem requiring integrating simulation models from cell-level phenomena all the way to macroscale developments (such as climate change or soil erosion).

The key inputs for the analysis are the Copernicus datasets, consisting of data from several satellites (Sentinel family) that produce radar and visible light data with up to 10m resolution. The radar data covers the whole globe every 2 days, with a monthly data rate of about 3PB. The visible light data covers the globe every 2-5 days, with a monthly data rate of about 4.5BP.

The availability of such fine-grained, global time series makes it possible to correlate and validate different Earth System Models that simulate the combined impact of several mechanisms with complex interaction patterns. The use case will use one such modelling system - the PROMET software - as a validation tool for services that provide an efficient and semantically rich interface to Copernicus data sets.

### 6.1 Copernicus use case motivation

Global Change, which subsumes the accelerating influence of humans on the natural environment, can be experienced in manifold ways. The most prominent are climate change, land use change including changing land management intensities, changing demand for biomass resources, increasing inputs of nutrients, changing mass balances of glaciers and changing global cycles of important elements like carbon and nitrogen. These changes influence the availability, quality and allocation of water resources, the productivity of the biosphere, the intensity of land cultivation and land use on all scales from local to global and will force to adapt to changing future boundary conditions.

With increasing area, the characteristic land surface processes and features, like snow dominated mountain processes, intensive agriculture and irrigation, land use patterns and geological settings simultaneously influence the overall reaction of an area or watershed. It is therefore necessary to treat this large variety of land surface processes and human interventions in a consistent manner within one model.

Therefore this use case analysis earth observation data as a spearhead activity for piloting and stress-testing advanced data analysis techniques that can be used to select and preprocess specific (e.g. spatially limited) time-series from the Copernicus datasets that are rapidly approaching exascale level.

### 6.2 Goal of the Copernicus use case

The aim of the use case is to realistically simulate natural processes and impacts of human interventions based on a few rigorous principles, which ensure maximum predictive power. Among them is a strict conservation of mass and energy and no calibration using measured streamflow records. The used software has been developed for the last 25 years with the aim to provide an integrated, stable and versatile tool to simulate land surface processes and to consider the dynamic interactions between the different land surface compartments and the human influences in the processes.

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.

### 6.3 Workflow of the Copernicus use case

The use case is based on a tightly coupled modelling framework PROMET (Processes of Mass and Energy Transfer) that combines modelling the macroscale 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. This use case combines tools for:

- Modelling a wide variety of global, local and cellular phenomena
- Simulation of the system-level combined impact of these processes
- Basic pattern recognition tools, e.g. adaptive and Bayesian methods for social media analysis and for data mining from the Copernicus datasets. Deep learning to identify human structures and their changes over time being considered
- Verification approach that allows adjusting simulation parameters based on the actual, observed development in the earth observation data during the simulation process.
- Dimension reduction approaches to manage Copernicus datasets (multi-sensor, multi-temporal data-cube) more efficiently

The Use Case initially starts (see Figure 9) by accessing data sets from the Copernicus observations. These data are fetched via an adapter, which needs to be implemented during the project. The fetched data sets will be stored in the PROCESS storage facilities. It includes raw and metadata.

With a newly developed preprocessor, some of the data will be preprocessed on the PROCESS computing resources. This will maximize the parallelization degree and enlarge the possible processed amount of data within a given time frame compared to the actual processing. The processed data is again stored within the PROCESS storage resources.

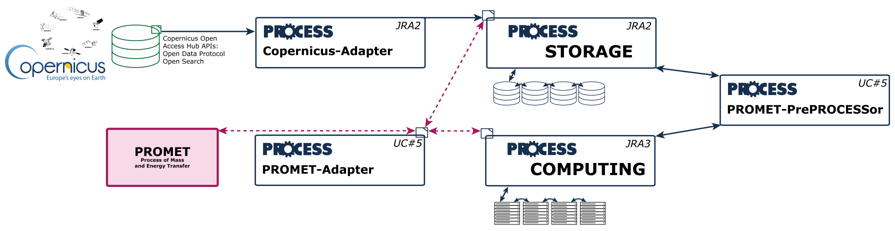
Copernicus
Europe's eyes on Earth
Copernicus Open Access Hub APIs: Open Data Protocol Open Search
PROCESS JRA2 Copernicus-Adapter
PROCESS JRA2 STORAGE
PROCESS UC#5 PROMET-PrePROCESSor
PROMET Process of Mass and Energy Transfer
PROCESS UC#5 PROMET-Adapter
PROCESS JRA3 COMPUTING

**Fig. 9** Workflow scenario.

The closed-source PROMET will fetch this data via a new so-called PROMET-Adapter, which can grant access to the data storage and also grant access for the PROMET software to be executed on the PROCESS computing resources.

## 7 Common architecture driven by the use cases

### 7.1 Architecture components

The PROCESS project bases its developments on the architectural framework model presented in Figure 10. This is a functional model, helping to align developments in each of the following “service component clusters”:

- Computing and storage infrastructure, responsible for optimising the performance of the underlying infrastructure while providing consistent interfaces to physical computing and data resources to the higher-level services
- SOA/Cloud solutions, supporting mapping of the “business processes” to mature, well-documented and supported software and service components
- Visualisation layer, supporting discovery and investigation of the input datasets, data pipeline design, scripting, as well as visualisation and further processing of the results of the analysis.

The main challenges in processing of exascale data which we want to tackle in PROCESS are as follows: extreme data sizes and data structure complexity. The former often prevents migration of data to a remote computation location, while the latter requires extensive domain knowledge in order to be able to use data effectively [3]. This forces users of exascale data to perform computations in situ, and only the largest research institutions and the best funded projects can afford to acquire access to an infrastructure able to house both the data and the extreme computational resources that can process it. Additionally, the expert domain knowledge required to be able to make sense of the data is often hard to find. These requirements make exascale processing prohibitively expensive for smaller players.

Designing an architecture which alleviates some of these problems and brings exascale data closer to smaller research organizations, and even to individual researchers is a crucial step in fulfilling the objectives of PROCESS. The architecture must be able to decouple access to the data from their use in computation and to provide simplified access to individual components of the data.

In PROCESS, we will also design a reference architecture (in JRA1) that not only addresses exascale data management challenges related to infrastructure owned by the consortium but also deals with data management challenges across existing Research Infrastructures (RI). RIs play an increasingly important role in the advancement of knowledge and technology in Europe. They are a key instrument in bringing together stakeholders with different backgrounds to look for solutions to many of the problems which the exascale data society is facing today.

Research infrastructures (RIs) offer unique research services to users from different countries, attracting young people to science, and helping shape scientific communities. RIs are hosted at research institutes, use a combination of high-performance computing (HPC), grids and clouds, and are often interconnected with high-capacity networks. RIs are accessed by the scientific community through a layer of high-level services that efficiently and transparently manage the execution of complex distributed CPU and data-intensive applications. Often these applications, modeled as workflows, are composed by loosely coupled tasks or jobs which communicate through file exchange systems [1, 21]. Such file exchange is facilitated by Distributed File Access Services (DFASs) which offer a single point of entry to discover and manage files and replication to improve file availability. Despite the effort to move computation to the data, there are cases where this is simply not possible. Technical constraints such as privacy and security issues, lack of computing power or the need for specialized hardware prevent computation from reaching the data [6, 20].

Typically, scientific workflows need access to datasets which are processed to generate new datasets, which have to be further processed by subsequent tasks to achieve a defined goal [17]. Therefore any DFAS which supports such an execution model needs to maintain strict consistency throughout its file access points. To enable use of off-the-shelf software and to hide the complexity of the network from the application developers and end users some DFASs offer standardized protocols [16, 2, 18] which decouple the development of client software from the DFAS, enabling implementation of clients that can present the DFAS as a POSIX file system through network transparency. Network transparency hides the way protocols transmit and receive data over the network; thus any operation that can be performed on a local file can also be performed on a remote file. To scale with increasing request load, Distributed File Access Services (DFASs) often employ redundant pools of servers. Despite the availability of RIs, DFASs do not always take advantage of their capabilities. It is rarely the case when DFASs interact with network devices (switches, routers, etc.) to optimize data transfers and maintain a quality of service (QoS). The main hurdle for interacting with network devices to optimize data transfers is the configuration of these devices, each of which relies on different protocols and interfaces. This makes large data transfers between RIs difficult to facilitate and slows down the life cycle of scientific applications [5, 10, 11].

From the point of view of computations, the focus of the project will be to facilitate simulation, modelling, and development of mathematical method and tools for exascale data processing. This, in particular, will include:

- services that enable execution of simulations models as interactive workflows using heterogeneous software, including modern languages for scientific computing
- services that facilitate access to HPC and cloud resources by using well-defined standards
- services that facilitate programming for multicore architectures
- benchmark and monitoring services

According to the requirements analysis, the main demands placed on the architecture are: an exascale data-capable, service-oriented, and cloud-based architecture allowing exascale computational research even for organizations which do not possess extreme resources. Considering the demands, a modular open-source architecture is proposed (Figure 10). It is divided into three main modules: (1) exascale data module led by work package 5 (green boxes), (2) exascale computing module led by work package 6 (red boxes), and (3) service orchestration module led by work package 7 (blue boxes, it includes a user interface).

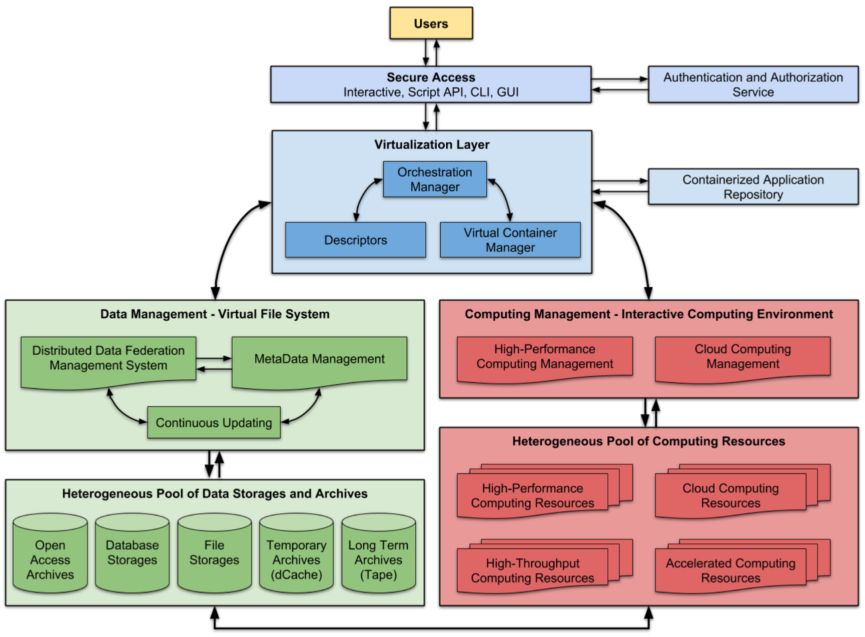
Users
Secure Access
Interactive, Script API, CLI, GUI
Authentication and Authorization Service
Virtualization Layer
Orchestration Manager
Descriptors
Virtual Container Manager
Containerized Application Repository
Data Management - Virtual File System
Distributed Data Federation Management System
MetaData Management
Continuous Updating
Computing Management - Interactive Computing Environment
High-Performance Computing Management
Cloud Computing Management
Heterogeneous Pool of Computing Resources
High-Performance Computing Resources
Cloud Computing Resources
High-Throughput Computing Resources
Accelerated Computing Resources
Heterogeneous Pool of Data Storages and Archives
Open Access Archives
Database Storages
File Storages
Temporary Archives (dCache)
Long Term Archives (Tape)

**Fig. 10** The initial functional design of the PROCESS architecture [4].

Users access the architecture through an application-oriented scientific gateway which provides secure access in the form of an interactive environment. The environment is represented as a script application programming interface (API), command-line interface (CLI), and graphical user interface (GUI). Authentication and authorisation of the user is performed by a dedicated service.

In order to alleviate the problems with processing exascale data, each of use case scenarios will be expressed as a set of containers or virtual machines. Commonly, workflow/scenario steps consist of applications that have enormous dependencies. This approach will unlock the full potential of the infrastructure, and also provide a secure and isolated environment. However, it implies virtualization/containerisation of an application environment as well as its execution environment. The architecture will provide a containerized application repository, which will offer a container or virtual machine per workflow/scenario step.

The core of the platform services is the virtualization layer. This component will be responsible for the deployment of application services. Once the services are deployed, they will be orchestrated by high-level descriptions automatically. This element will be responsible for resource allocation, configuration of all services and management of their lifecycle.

To reach an exascale-capable infrastructure, interoperability across multiple computing centres is essential. The whole infrastructure is divided into two parts: (1) an extreme large data service-oriented infrastructure, and (2) an extreme large computing service-oriented infrastructure. The first part is modelled as a distributed virtual file system (DVFS), an abstraction layer on top of a more concrete file system. It allows applications to access different types of file systems (within a data storage federation) in a uniform way. It also supports access to files from multiple hosts via a computer network. The main requirements are to fit the federated architecture principles and to avoid forcing users to rely on any specific client. DVFS also includes a manager for metadata and supporting data services (such as monitoring).

The second part is responsible for management of computing resources. It is modelled as an interactive computing environment, allowing to access different computing resources in a uniform manner. It supports extreme large computations that require heterogeneous infrastructures. It offers HPC computations, HTC computations as well as cloud computations, and supports accelerators.

## 7.2 The initial technology-based architecture of the PROCESS project

Integration[9] as a whole is depicted by Figure 11. Users will interact with the architecture through a Model Execution Environment (MEE)[10]. The whole architecture is driven by a SOA approach, and so RESTful interfaces are dominant. MEE will provide authentication and authorisation services. It will be integrated with

[9] The subchapter provides an example of a technology-based architecture that meets the requirements of the use cases (presented above) as well as the conceptual and functional requirements derived from the functional model of the PROCESS architecture.

[10] Details concerning the MEE can be found at `http://www.cyfronet.krakow.pl/cgw17/presentations/S7_2-EurValve-MEE-Oct-2017-v02.pdf`

Jupyter[11] and offer browsing of files through LOBCDER[12]. MEE will be connected to a TOSCA[13] template repository through SSL, and Cloudify[14] through a REST API. Following successful authentication and authorisation, users will be able to specify a workflow. Cloudify will obtain its description in a TOSCA template and deploy dedicated Docker/Singularity images on corresponding infrastructure elements. In order to achieve this functionality, Cloudify will include three plugins: (1) the LOBCDER plugin, (2) the Rimrock plugin[15], and (3) the Atmosphere plugin[16], exploiting the underlying REST API (and WebDAV, within LOBCDER plugin). HPC resources will be managed by Rimrock through the GSI-SSH protocol, and cloud infrastructures will be governed by Atmosphere. The distributed virtual file system will be provided by LOBCDER. It will provide direct access to data sources through a dedicated virtual file system driver. However, if the service is complicated (for example data filtering, data integration etc.), then it will go through the DISPEL Gate[17]. LOBCDER is also integrated with DataNet which will be responsible for management of metadata. More details about the technologies are provided in the next section, which focuses on the current state of the art and a description of the relevant tools.

## 8 Conclusion

The chapter describes 5 exascale use cases: exascale learning on medical image data, exascale challenges within the LOFAR data volumes, supporting innovation based on global disaster risk data, ancillary pricing for airline revenue management, and agricultural analysis based on Copernicus data. These use cases come from academic and industrial sphere with the following main requirements:

**Medical use case**: exascale computational processing methods that are able to exploit accelerated computing (access to GPUs, ability to dynamically allocate tens of TBs of storage that is co-located with the GPU clusters)

**SKA use case**: scalable workflows that are able to cooperate with exascale datasets (ability to deal with large variations in the data access times – several order of

[11] Julpiter webpage http://jupyter.org/

[12] LOBCDER2] LOBCDER webpage https://ivi.fnwi.uva.nl/sne/wsvlam2/?page_id=14

[13] OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA)Âăhttps://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca

[14] Cloudify] Cloudify: Cloud & NFV Orchestration Based on TOSCA https://cloudify.co/

[15] RIMROCK webpage https://submit.plgrid.pl

[16] Atmosphere webpage http://dice.cyfronet.pl/products/atmosphere

[17] Atkinson, M., Brezany, P., Krause, A., van Hemert, J., Janciak, I., Yaikhom, G.: DISPEL: Grammar and Concrete Syntax, version 1.0.The Admire Project, February 2010. Accessed March 2011.Âăhttp://www.Admire-project.eu/docs/Admire-D1.7-research-prototypes.pdf

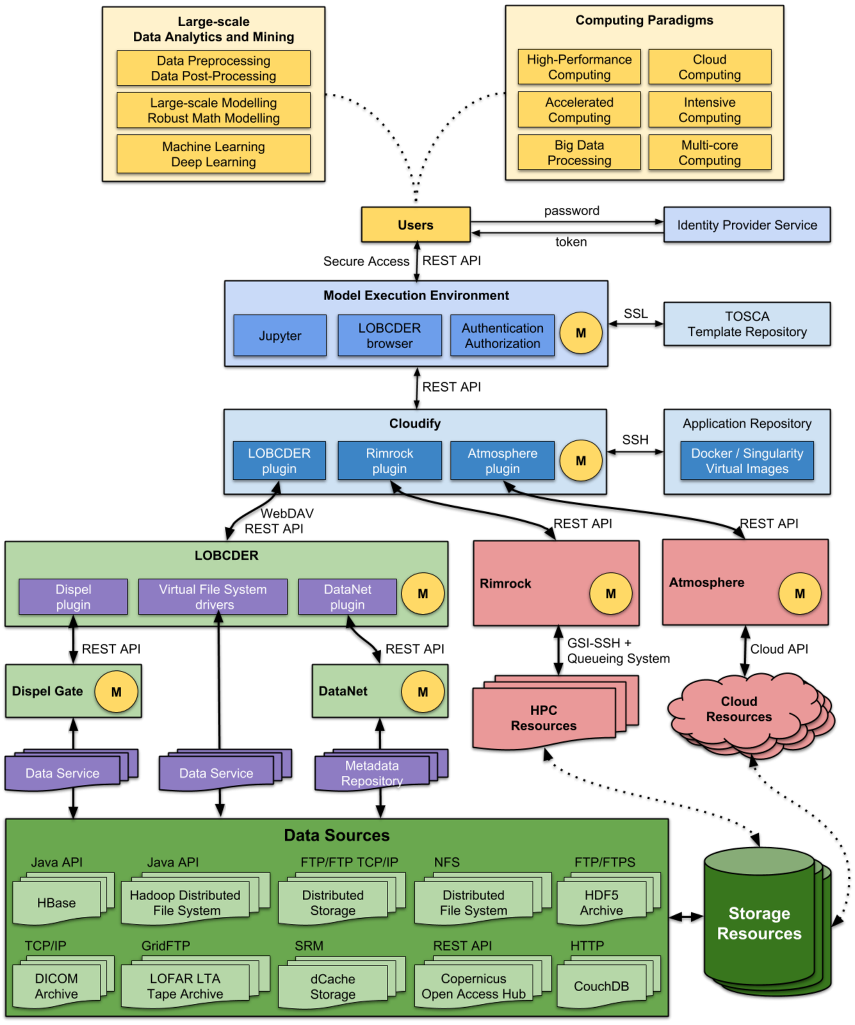

**Fig. 11** The technology components of PROCESSâĂŹ architecture (the yellow circle with the letter M represents a monitoring agent from Zabbix).

magnitude differences in access latencies)

**Ancillary pricing use case**: : responding to a large number of requests within milliseconds per request, data confidentiality, financial pressures, real-time

**Copernicus use case**: methods suitable for scalable data extraction (PB-scale "smart cache" with preprocessing capabilities)

The last section is dedicated to designing of the PROCESS architecture, with a high-level structure of the project platform. According to the requirements, the architecture is based on containerization and virtual machines supported by an exascale capable distributed virtual file system, and computing manager. This approach will utilize the available infrastructure with minimal overhead, and provide a platform capable of satisfying the requirements of the use case communities.

In the near future, we will investigate a potential extension and improvement of the containerization approach. We are planning to study the management of the PROCESS architecture sub-components. Currently, the best solution candidate is a service-driven micro-architecture. It will make the PROCESS architecture more flexible and scalable, however, further examinations and analyses are needed which will be the subject of our next research.

## Acknowledgement

This work is supported by projects EU H2020-777533 PROCESS PROviding Computing solutions for ExaScale ChallengeS, APVV-17-0619, and VEGA 2/0167/16.

## References

1. R. Agarwal, G. Juve, and E. Deelman. Peer-to-peer data sharing for scientific workflows on amazon ec2. In *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*, pages 82–89. IEEE, 2012.
2. B. Allcock, I. Mandrichenko, and T. Perelmutov. Gridftp v2 protocol description. *GridFTP Working Group, Tech. Rep*, 2005.
3. 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.
4. 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*, pages 461 – 468. IEEE, 2018.
5. J. Chen, A. Choudhary, S. Feldman, B. Hendrickson, C. Johnson, R. Mount, V. Sarkar, V. White, and D. Williams. Synergistic challenges in data-intensive science and exascale computing: Doe ascac data subcommittee report. 2013.
6. L. B. Costa, H. Yang, E. Vairavanathan, A. Barros, K. Maheshwari, G. Fedak, D. Katz, M. Wilde, M. Ripeanu, and S. Al-Kiswany. The case for workflow-aware storage: An opportunity study. *Journal of Grid Computing*, 13(1):95–113, 2015.
7. F. Dottori, P. Salamon, A. Bianchi, L. Alfieri, F. A. Hirpa, and L. Feyen. Development and evaluation of a framework for global flood hazard mapping. *Advances in Water Resources*, 94:87 – 102, 2016.
8. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

9. 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*, pages 281–314. Elsevier, 2018.
10. M. Kluge, S. Simms, T. William, R. Henschel, A. Georgi, C. Meyer, M. S. Mueller, C. A. Stewart, W. Wünsch, and W. E. Nagel. Performance and quality of service of data and video movement over a 100 gbps testbed. *Future Generation Computer Systems*, 29(1):230–240, 2013.
11. D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.
12. 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*, pages 2755–2763. IEEE, 2017.
13. G. Montavon, W. Samek, and K.-R. Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 2017.
14. R. Rudari, F. Silvestro, L. Campo, N. Rebora, G. Boni, CIMA Research Foundation, C. Herold, and UNEP-GRID. Improvement of the global food model for the gar 2015. *Global Assessment Report on Disaster Risk Reduction 2015*, 2015.
15. 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*, 598:A104, 2017.
16. A. Sim and A. Shoshani. The storage resource manager interface specification, version 2.2. In *CERN, FNAL, JLAB, LBNL and RAL*. Citeseer, 2007.
17. A. Simonet, G. Fedak, and M. Ripeanu. Active data: A programming model to manage data life cycle across heterogeneous systems and infrastructures. *Future Generation Computer Systems*, 53:25–42, 2015.
18. Storage Networking Industry Association. Cloud data management interface (cdmi). 2010.
19. 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*, pages 1–9, 2015.
20. R. Tudoran, A. Costan, O. Nano, I. Santos, H. Soncu, and G. Antoniu. Jetstream: Enabling high throughput live event streaming on multi-site clouds. *Future Generation Computer Systems*, 54:274–291, 2016.
21. L. Wang, J. Tao, R. Ranjan, H. Marten, A. Streit, J. Chen, and D. Chen. G-hadoop: Mapreduce across distributed data centers for data-intensive computing. *Future Generation Computer Systems*, 29(3):739–750, 2013.