This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 777533.

PROviding Computing solutions for ExaScale ChallengeS

D8.1	Validation Report of 1st PROCESS update		
Project:	PROCESS H2020 – 777533	**Start / Duration:**	01 November 2017 36 Months
Dissemination[1]:	Public	**Nature[2]:**	**R**
Due Date:	31 July 2019	**Work Package:**	**WP 8**
Filename[3]	PROCESS_D8.1_ValidationReport_v1.0.docx		

ABSTRACT

Deliverable D8.1 aims to continue to report on the progress of the PROCESS five use cases, D8.1 builds up on the scenarios defined in the deliverables D4.1 and D5.2 and the initial trials and tests presented in Deliverable D4.3. To the contrary to our previous deliverables where we focused mainly on the two mature use cases, in D8.1 we report the progress of all five PROCESS use cases describing their current status and the integration plan with the PROCESS infrastructure as a whole though concrete scenarios showing how they are progressing towards their final goal planned at the end of the project.

[1] PU = Public; CO = Confidential, only for members of the Consortium (including the EC services).

[2] R = Report; R+O = Report plus Other. Note: all "O" deliverables must be accompanied by a deliverable report.

[3] eg DX.Y_name to the deliverable_v0xx. v1 corresponds to the final release submitted to the EC.

Deliverable Contributors:	Name	Organization	Role / Title
Deliverable Leader[4]	Adam Belloum	UvA	Deliverable coordinator
Contributing Authors[5]	Reginald Cushing, Onno Valkering, Adam Belloum	UvA	Writers
	Souley Madougou, Jason Maassen	NLESC	Writers
	Maximilian Höb, Jan Schmidt	LMU	Writers
	Balázs Somoskői	LSY	Writer
	Mara Graziani, Henning Müller	HES-SO	Writers
Reviewer(s)[6]	Ondrej Habala	UISAV	Reviewer
	Matti Heikkurinen	LMU	Reviewer
Final review and approval	Höb, Maximilian	LMU	Coordinator

Document History

Release	Date	Reasons for Change	Status[7]	Distribution
0.0	01.06.2019	Table of content	Draft	All
0.1	10.06.2019	Section 3	Draft	All
0.2	01.07.2019	Section 2, and Section 3	Draft	All
0.3	13.07.2019	First draft all sections	Draft	All
0.4	15.07.2019	First complete draft	In Review	All
0.5	26.07.2019	Revision ready for submission	In Review	All
1.0	30.07.2019	Final Version	Released	All

[4] Person from the lead beneficiary that is responsible for the deliverable.
[5] Person(s) from contributing partners for the deliverable.
[6] Typically, person(s) with appropriate expertise to assess the deliverable quality.
[7] Status = “Draft”; “In Review”; “Released”.

Table of Contents

Executive Summary

In Month 15, we released the alpha version of the PROCESS data services (D5.2), and the performance model to be used to assess the PROCESS infrastructure (D3.4). After some initial trials and tests aiming to test data services with the two mature use cases UC#1 and UC#2, we have revised all the use case requirements in D4.3. As a follow up of the process of validation of the PROCESS infrastructure, we describe in this deliverable the new experiments that have been performed to further test the PROCESS infrastructure with use cases scenarios derived from the 5 PROCESS use cases.

In this deliverable, we present more results of the scenarios derived from most mature use cases namely UC#1 and UC#2. As the implementation of the use cases is still a work in progress, we describe the scenarios that are currently being developed and tuned to take advantage of the released PROCESS services. As a result of the programmable micro-infrastructure implemented in PROCESS, the generic services currently developed for data transfer and being tested with UC#1 and UC#2 will be used by other use cases.

List of Figures

List of Tables

1 Overview

In deliverable D4.3, we have revised the application requirements in the light of the new insight we have acquired by developing both the PROCESS infrastructure and the use cases themselves. Comparing to the initial requirements listed in Deliverable D4.1, we did not identify any major deviation from our original requirements from both software and hardware (computing and storage resources) points of view. Besides some minor changes and adaptations related to the use of certain technologies (Singularity vs Docker), or supporting different protocols for data transfer over WAN (SCP, GridFTP, ...), we pointed out[8] that we might need to set up special Data transfer nodes (DTN) to speed up the stage the data in/out in the case where storage and data resources are not co-located (example is the LOFAR archive which is distributed over The Netherlands, Germany, and Poland).

In this deliverable we will describe scenarios derived from the use cases to test how much the current PROCESS infrastructure satisfies the new requirements, and to identify what still needs to be done in the coming months to completely address all the new requirements. The selected use case scenarios are quite similar to the to the final use cases and will help us to move one step toward our final targets. We will also use this opportunity to partly port the use cases to the PROCESS infrastructure.

In the following sections, we present 5 scenarios related to the five use cases considered in PROCESS.

2 UC#1: Exascale learning on medical image data

Most of the new requirements for UC1 were met, namely:

1) The integration between Docker containers and Singularity was made available with the possibility to switch from one solution to the other with flexibility. This allowed the porting of experiments to the computing centres using Singularity. Time and memory requirements were tested in a configuration similar to the final use case;
2) The requirement of keeping the Python libraries up to date is met by integrating a new Docker container with the UC1 software in the infrastructure once a new release is available. For instance, updates of the Keras and Tensorflow libraries can be met by building an updated version of the container and deploying it on the infrastructure.
3) The SCP protocol for data transfer is currently available with data staging service.

The support of Uber's Horovod tool[9] inside the Singularity container has yet to be deployed and tested in the different computing centres.

The use case application is organized into three layers. Layer I implements the extraction of patches of dimensions 224x224 pixels from the gigapixel slides of breast lymph nodes tissue. Patches are random sampled from the slide, in which areas of tumour were annotated by a physician. Patches belonging to a tumorous region are assigned a 'tumor' label (a Boolean variable equals to True). The extracted data are stored in an intermediate dataset with the corresponding labels. Layer II loads the intermediate dataset of patches and labels and trains a state-of-the-art deep convolutional network to classify the two patch types. Different models can be chosen by a configuration parameter. Layer III focuses on network robustness and interpretability. A summary of the use case application layers can be found in Table 1.

8 in Deliverable D4.3 page 9

9 The goal of Horovod is to make distributed Deep Learning fast and easy to use. The primary motivation for this tool lies in its deployment, which has been facilitated to transform a single GPU TensorFlow application into a multi GPU distributed architecture. See also: https://eng.uber.com/horovod

Table 1: UC#1 internal structure and processing flow

Layer I: Data pre-processing and patch extraction	**Layer II: Local and distributed training**	**Layer III: Performance boosting and interpretability**
Creation of normal and tumour tissue masks from the physician's annotations	State of the art deep convolutional networks currently implemented: Resnet50, Resnet101, InceptionV3	Generation of intermediate visualizations
Random sampling of high-resolution patches and labelling	Local training on single and multiple GPUs	Feature importance analysis
Intermediate storage of the patches on H5DS	Training on HPC clusters	Perturbation robustness analysis

In the next sections, we report the results of the use case in three production scenarios as defined in D3.1, page 10.

Scenario 1, mostly focuses on the use case requirements and performances for the single computing site. Performances are compared across computing centres in Section 2.1.a. Scenario 2 evaluates the speed up performances of running several containers on a single computing site (see Section 2.1.b). Scenario 3 highlights the bottlenecks and limitations that need to be faced to run several containers on several computing sites. Especially, data transfer is analysed in Section 2.1.c.

2.1 UC#1 scenario:

2.1.1 Scenario 1: a single container of UC1 runs on one computing site

Benchmarks were computed on 3 of the 5 computing sites for Layer I and Layer II of the UC#1 software. The use case application handling of the resources and access to CPU and GPU memory was intensively optimized, thus overcoming the resource exhaustion issues reported in D2.1. Table 2 and Table 3 illustrate the current status of the application layers I and II. First order statistics of computational requirements for Layer I are reported in Table 2. First order statistics for the time requirements of Layer II are reported in Table 3. Performance of Layer II is also reported in Table 3 in the form of model accuracy.

Table 2: Performance measurements of UC#1, Layer I

Resource location	**Hes-so**	**UvA**	**AGH**
patch sampling time [s/patch]	0.41+-0.1	N/A	0.5+-0.1
Data loading time [ms/patch]	2.0	1.5	0.5

Table 3: Performance measurements of UC#1, Layer II

Resource location	**Hes-so**	**UvA**	**AGH**
Training accuracy	96.91+-0.45	96.1+-0.24	84.3
Validation accuracy	85.6+-6.2	83.1	93.7
Training time [s/epoch[10]]	2440.80	1203.68	17277
10 epochs time [h]	7	3.34	47

The evaluation of performances highlights very efficient data extraction and loading at AGH, with only 0.5 ms to load a single patch. The high-performance GPUs available at UVA, by contrast, provide fast computations on a single GPU, halving the computational time of the model training (from 7 hours to 3.34 hours).

2.1.2 Scenario 2: run several containers on a single computing site.

An intense refactoring of Layer I was performed to optimize access to the resources and deploy the use case on the HPC infrastructures. Cyfronet in Krakow, Poland, has been chosen to benchmark the parallel execution of several containers of the use case application Layer I. Two different parallelization methods were analysed:

1) A batch of patches is extracted for the single patient. All processes extract patches with the same SLURM seed.
2) A batch of patches is extracted with random sampling for all the patients in the dataset. Each process loads the data for all patients and extracts patches in a set of random locations specified by a SLURM seed.

Statistics were computed on a subsample of 5 patients with 10 CPUs. A total number of 5000 patches was extracted in less than 5 minutes for both methods. Single-CPU requirements were averaged and shown in Table 4.

Table 4: Memory usage and computing time at AGH.

	Memory Utilized	CPU time RAW [s]	Max Job Wall-clock time
AGH extraction (random sampling)	1.7 Gb +- 0.26	263.3+-22.9	00:04:52

10 An epoch is an iteration of the learning process of deep neural networks, so increasing the number of epochs usually improves performances until saturation is reached.

Scaling is possible by increasing the number of CPU nodes and patients, as shown in Figure N. Ideally, by increasing the number of CPU nodes available from 10 to 1000 a number of 50 000 patches can be extracted in less than 5 minutes, with a linear speed-up.

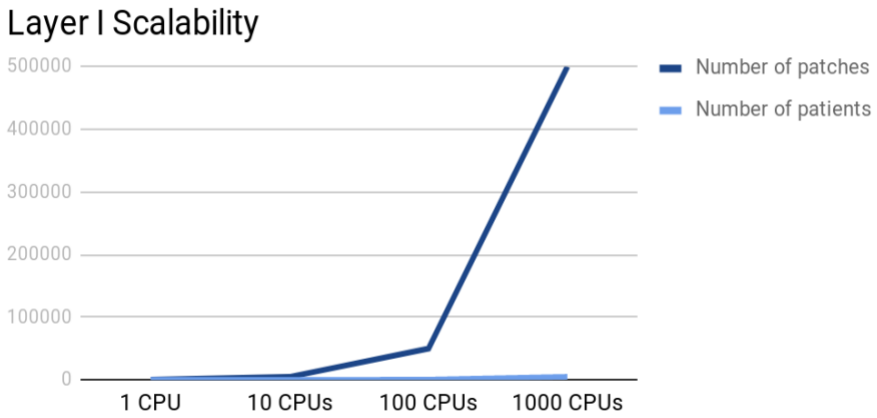

Figure 1: Showing the scalability of the system.

The parallelization of Layer II was initially tested on the local computing resources. Results highlight the possibility of at least a 20% of time save up by parallelizing the computations on 2 GPU NVIDIA K80, as shown in Table 5.

Table 5: Test results for running the system on a single rack server with one and two GPUs.

Resource	1x GPU K80	2 x GPU K80
training time [s/epoch]	2440.80	2004.3

2.1.3 Scenario 3: run several containers on several computing sites.

The results from Scenario 1 and Scenario 2 suggest the ideal configuration for Scenario 3 of running Layer I at AGH to best profit of the efficient data handling and subsequently transfer the data from AGH to UvA to execute Layer II. The time for data transfer could constitute a possible bottleneck.

2.2 Testing site-to-site staging

Table 6: Tests with data transfer between different locations.

protocol - SCP	30GB Camelyon16.partAA - MB/s	30GB Camelyon16.partAB - MB/s	30GB Camelyon16.partAC - MB/s	mean bw - MB/s	duration - minutes
LRZ-V-DTN to AMS-DTN	86.8	90.8	84.9	87.5	5.85
LRZ-V-DTN to AGH	33.1	31.2	32.2	32.17	15.92
LRZ-V-DTN to LISA	25.5	64	29	39.5	12.96
AMS-DTN to LISA	167.9	172.6	166	168.83	3.03
AMS-DTN to AGH	53	53.9	38.9	48.6	10.53
LISA to AGH	40	21.3	29.5	30.27	16.91

Table 6 shows cross site staging part of the Camelyon16 dataset using several protocols and strategies. Our initial approach is to use the widely available SCP protocol and use cluster head nodes to stage the data. From the table, this approach is shown to be one of the worst strategies e.g. LISA to AGH. Furthermore, head nodes are very unstable for staging such data with frequent stalls and broken connections. A second strategy is to use Data Transfer Nodes (DTN). What we can show from the tests is that using such DTN nodes can speedup transfers, for example, transferring data from LRZ site to LISA is faster to use a DTN relay than to directly copy. E.g. LRZ-VDTN to AMS-DTN takes 5.85 minutes plus AMS-DTN to LISA is 3.03, cumulatively it is 8.9 minutes which is ~ 30% faster than a direct copy which is 12.96 minutes. The results also show the added speedup by using campus networks. The AMS-DTN and LISA are on campus thus their bandwidth is approximately 4 times better than the rest. Using DTNs as cache staging nodes could potentially accelerate data transfers between sites. With these tests we feel that we have a basis for the upcoming steps for UC1. Moreover, the additional requirement of SCP protocol for data transfer as stated in D4.3 page 9 has been met. This ensures that data transfer can be applied to different types of users, and especially hospital institutions which may not have open FTP access for security reasons.

3 UC#2: Square Kilometre Array/LOFAR

Although UC2 could meet most of its requirements as of D4.3, some components have not been implemented yet.

The astronomer-friendly Web user interface (WUI) is implemented as a Web application in cooperation[11] with the EOSC pilot [12]with added functionality to run the UC#2 pipeline. The actual pipeline, illustrated in Figure 2 below, was already containerised but was (and still is) incomplete as it did/does not include the direction-dependent (DD) calibration part of the DD pipeline. For the latter, UC#2 has various types of algorithms to choose from. These various

11 The developer of the UI is a colleague at NLeSC with whom we collaborate. The EOSCpilot itself being an EU Horizon 2020 project, it is recommended to reuse its end-products within PROCESS.

12 https://eoscpilot.eu

algorithms can be implemented in their own containers and then plugged-in to the pipeline. It was initially planned to use DDF, which in current state requires special high-end nodes. However, parallel or easily parallelizable algorithms for DD calibration, such as SAGECaL or FACTOR, are readily available and would make the special high-end unnecessary.

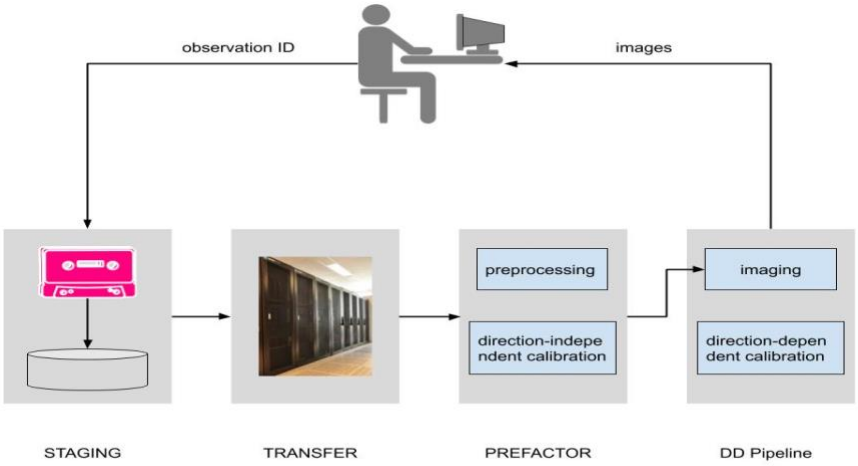

Figure 2: UC2 pipeline implementation status. Currently, only DD calibration is missing.

The integrated services were computing services and a “custom” stage-out data service. The observation data were manually moved to the computing site. Upon completion of the pipeline, the resulting FITS images were transferred to the UI server for conversion to JPEG and visualisation. In the current update, actual PROCESS data services for both stage-in and data transfer to HPC have been integrated. As can be seen from the image in Figure 3 (updated from page 19 of D5.2), UC2 is using two more data services compared to the first prototype: the “LOFAR LTA observation staging” service for moving observational data from the tape archive to a temporary location and the "sshfs storage adaptor" container for transferring them from that location to HPC for computation. The protocol used for the transfer is the storage resource management (SRM), which is just marginally less efficient than GridFTP. This functionality can be used for any combination of LTA archive and computing site within PROCESS infrastructure. Observations from the three LTA archives are already being used for staging and benchmarking.

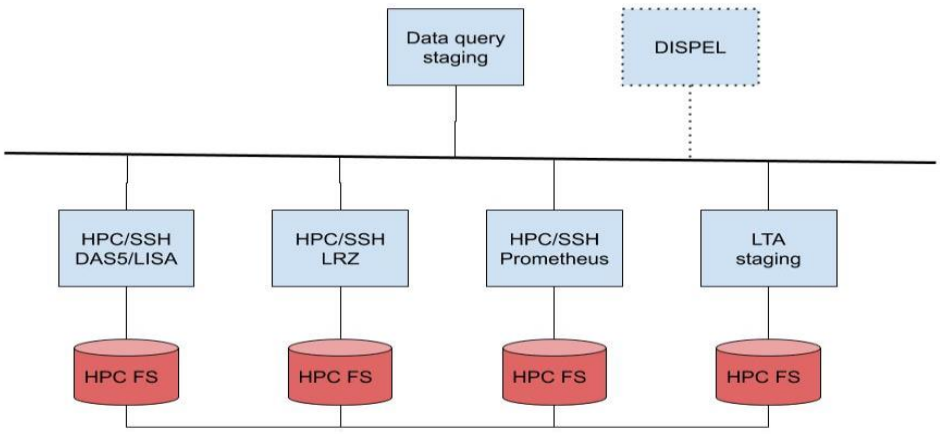

Figure 3: Current use of PROCESS services by UC#2

It is also planned to soon use two other services, namely a DISPEL adaptor for image conversion and compression, and the GridFTP delegation copy service along with the data transfer nodes (DTN) for faster transfer. As for the computing site, while a version of UC#2 was already running on Prometheus at CYFRONET, the more advanced one is tested and benchmarked only at DAS5, the Dutch-based PROCESS computing site. The next step is to run the pipeline on more than one computing site across two or more countries.

Table 7: UC#2 requirement implementation status

Requirement	Status in D4.3	Current status
Special computing nodes	Available on Prometheus but not tested	parallelisation and/or use of parallel DD calibration tools under consideration
Efficient data transfer using GridFTP or others	GridFTP support under development	Still under development along with use of DTNs
Horizontal scaling	Multi-site support under development	Functionality built in Xenon and Xenon-flow, but needs testing

3.1 UC#2 Scenario:

In this section, we use the actual use cases to evaluate the platform scalability through the scenarios defined in section 2, "Identification of measurands" of D3.1 page 10 and recalled here.

- Scenario 1: consists in running one UC container on one computing site;
- Scenario 2: several containers are run on one computing site;
- Scenario 3: run several containers on several computing sites.

Among the 8 defined measurands identified in D3.1, we only collect those contributing to PROCESS overhead and those pertaining to staging and data transfer to the computing sites. For the latter, we separately evaluate the queueing and preparation time due to use of tape robot technology for the archive, the actual stating and the data transfer to the HPC sites. We use a single small data set to estimate the queueing overhead and, increasingly bigger data sets (from 20GB to 320GB, by 2x) for the staging and transfer.

For the platform overhead evaluation, every container runs a reduction pipeline for one observation (up to 16TB, but for the sake of time, we can use smaller datasets). Because we are focusing on the overhead, just as in deliverable D4.3, we are not interested in the actual computations. Consequently, each step in the pipeline involving computations is replaced with an execution of the validation container developed for validation in D4.3. For Scenario 2, we run consecutively 1, 2, 4, 8, 16 and 32 containers on one computing site. In Scenario 3, we plan to run consecutively 3, 6, 12, 24, 48 and 96 containers on three computing sites, i.e., the same number of containers as previously but per computing site.

3.2 UC#2 Benchmarks:

Data used in this use-case originates from the LOFAR Long Term Archive (LTA). The LTA makes use of three data centre locations: Amsterdam (NL), Jülich (DE) and Poznań (PL). Here, data is stored on tape drives, which provide stable long-term storage but aren't capable of direct access. To access the data, they have to be staged first. During this process a tape robot will retrieve the appropriate tape (s) and read the desired data to a cache. Thereafter, the data can be accessed directly from the cache. To gain insight into the overhead this introduces to the pipeline, we performed the following three benchmarks.

Estimating queuing and preparation time

The tape drives and robots at each of the LTA locations are not only shared by other LTA users, but also by other projects housed in the same data centre[13]. Therefore, it may be that a staging request will spend some time in a queue. In addition, the tape robot may need some preparation time before the data can be copied. To estimate this overhead, we staged a small file (< 100MB) so the reading time would be negligible compared to the remaining queuing and preparation time. We repeated this experiment ten times, at each LTA location, spread over different days, before averaging the recorded durations (Figure 4).

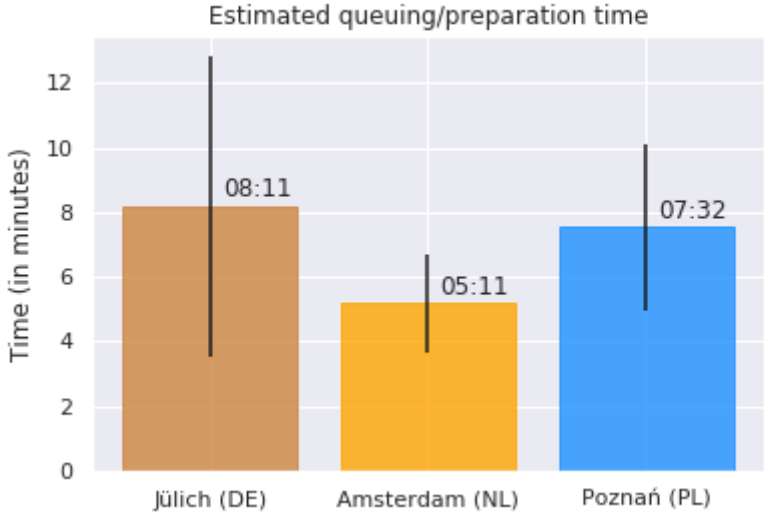

Figure 4: Estimated queueing and preparation time

The durations are quite variable, but in general can be placed in the five to ten-minute range. It might occur that queuing and preparation take longer, especially if multiple staging requests are filled simultaneously. However, we didn't experience this during our benchmarking period. The differences between the three locations can be attributed to (possible) different configurations and/or load at the respective data centres.

Total staging time as a function of total size

For an indication about the total duration of a staging request, we staged increasingly numbers of gigabytes (20GB, 40GB and 2x, 4x and 8x 40GB). We repeated these three times, at each LTA location[14], spread over different days before averaging the durations (Figure 5). We observe, again, that the durations are variable, but are reasonable. These durations are uncontrollable, as explained before, because of the shared nature of the LTA systems.

13 https://www.astron.nl/lofarwiki/doku.php?id=public:lta_howto#please_take_note_of_the_following

14 The LTA storage system in Jülich (DE) was undergoing maintenance at the time of the benchmarking, the results for this location will be added in the subsequent deliverable.

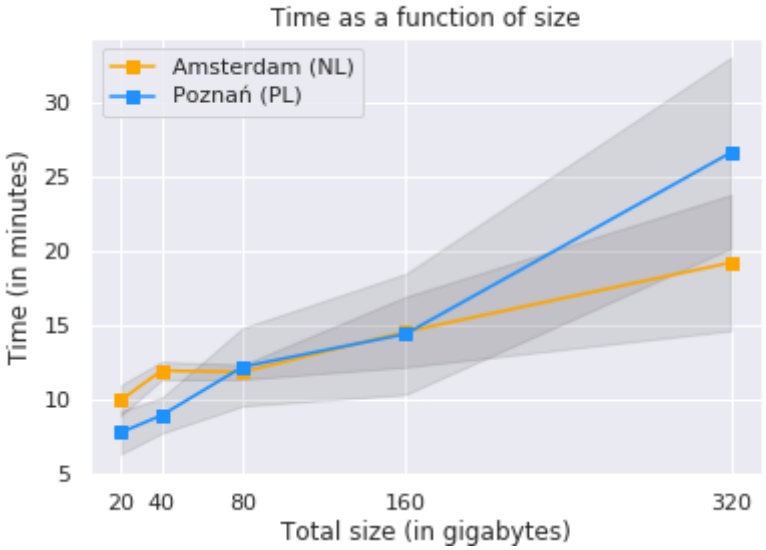

Figure 5: Duration of the staging in function of data size

Transfer speeds from LTA to HPCs

After the data have been staged, they can be transferred to a HPC cluster for further processing. We are benchmarking this transfer, to see to which extent this transfer of large data files across system boundaries induces a bottleneck in the overall use-case's pipeline. We measured the transfer speed (Figure 6) four times before averaging it, between each LTA location and the HPC clusters: LRZ in Garching (DE), LISA in Amsterdam (NL) and CYF in Kraków (PL). The transfers consisted of a single file of around 80GB and the transfer was limited to one hour.

The results show that the transfers between LTA locations and HPC clusters, with the exception of Amsterdam (NL) to LISA (NL), are suboptimal. For these transfers only public networks can be used. This results in transfer speeds in the range of roughly 5 to 12 MB/s. With these speeds a single 100GB file will take somewhere of 2 to 6 hours to be transferred from a LTA location to a HPC cluster. As a comparison, such a transfer over the fast connection between Groningen and LISA will only take about 6 minutes.

From these staging and transfer benchmarks, we conclude that the staging part doesn't necessarily imposes the biggest bottleneck on the use-case's pipeline. However, the transfer after the staging, from the LTA to a HPC cluster, will yield a bottleneck.

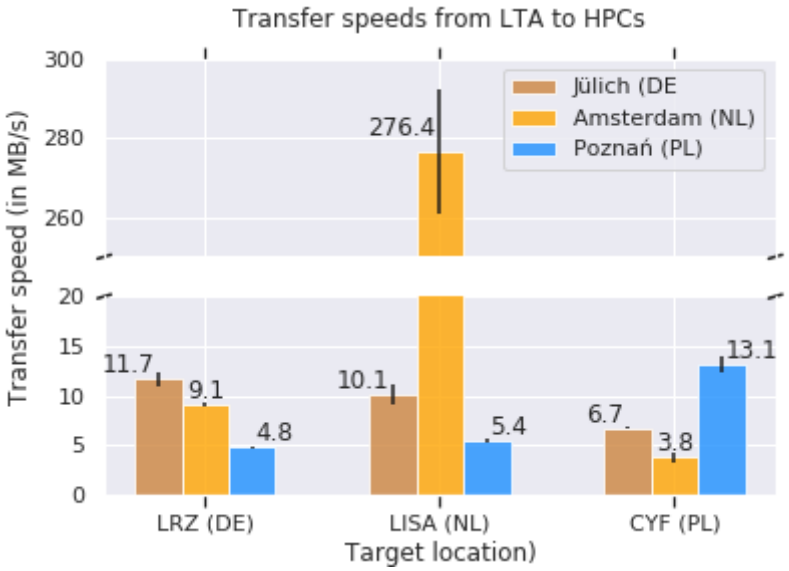

Figure 6: Transfer performance from LTA sites to HPC sites

Computing benchmarks

Unfortunately, given that the multi-site functionality is not implemented yet, we could only test scenarios 1 and 2. Furthermore, the peculiarities of UC#2 implementation necessitate an interpretation of the type of data being measured with relation to the measurands defined in D3.1, pages 8-10. For staging, the meaning of T3 and T8 stays unchanged. For measurands contributing to the overhead of PROCESS, just as figured out in writing D4.3 while running on Prometheus, T1 is not measurable or not useful for performance evaluation at the current state of UC#2 implementation neither. Neither is T4 as currently there is only one container to run. Consequently, the observed overhead is composed of only T2 and T7. T2 is coming from the use of a Web frontend to send requests for running containers to a server through HTTP, the latter requiring to specify the tasks to be run as CWL files which are transferred to the computing site. It then enqueues the tasks for another tool that prepares them for the HPC workload management system. Likewise, T7 is associated with the overhead for preparing and transferring standard error and output, and eventually other results, back to the UI system. Due to limitations in the current setup, we could only measure up to 32 containers running simultaneously on the computing site without errors. Finally, we are using the validation container for the computing steps as we are only interested in measuring the overhead.

In the figure below, we plot the behaviour of the two types of overhead, as well as the overall overhead, as the number of containers varies. We observe that T7 only increases slowly as shown in D4.3. T2 which is an aggregate of several types of overhead, also increases slowly up to 8 containers, but then increases substantially onward, which needs further investigation. The consequence of this is that the overall overhead also increases substantially after 8 containers and becomes almost linear in the number of containers, just as before (D4.3, section 3.1.2). The regression analysis shown on the right in the Figure 3.7 below, exhibits the relationship between the overall overhead (indicated by the letter *o*) and the number of containers (shown as *c*). Compared to our results from D4.3, we are doing worse because the slope of the regression line (shown in blue in the second plot and the equation at the upper-right) was 0.26 instead of the current 1.4. Clearly, this level of

overhead is not desirable as this will amount to about 42 minutes to entirely process the 1800 16TB-observations currently in the LTA. If there were "PASS/FAIL" levels, this will in between although it will negligible in the current situation where the astronomer has to wait for 4-5 days for just one observation. However, before taking defensive actions, the benchmarks will be run on a more stable and dedicated environment, probably from SURFsara. Compared to the previous evaluation conditions, both the software and hardware are different. The latter is experimental and frequently used by many other users and the former uses HTTP and CWL, the combined effect leading to more overhead relatively to previous setup. It might also be the case that we are uncovering more overhead than previously. To be checked in the next update.

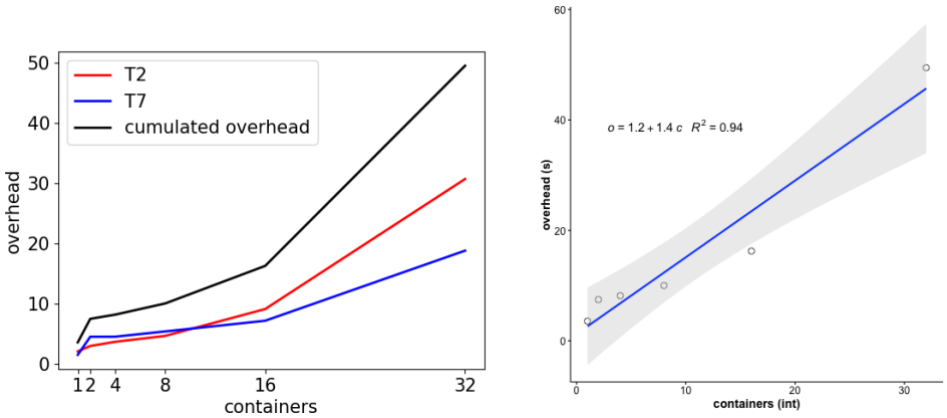

Figure 7: PROCESS platform overhead evaluation and modelling for current UC#2 implementation

4 UC#3: Supporting innovation based on global disaster risk data

The UC#3 has continued through four parallel activity streams:

1. Observing the behaviour of the user community of the currently published dataset
2. Seeking collaborative links with additional initiatives producing disaster risk data (and related datasets)
3. Using the existing contact network to engage with new stakeholder types that could benefit from access to disaster risk data or derived products
4. Analysing the PROCESS architecture to identify possible obstacles to and emerging opportunities from integration with the existing use case software

The outcomes of the first three activity streams are presented in the section 4.1 of this document. The architectural analysis resulted in the following observations:

- There are no feasible scenarios where the PROCESS platform couldn't support the basic functionality of the current data download portal.
- The emerging services of the PROCESS platform that are of particular interest are:
    - The ability to manage distributed/federated datasets using the data transfer nodes
    - The seamless access to data that may reside on tape archives by reusing UC#2 components
    - Using data made accessible through the portal as input in a PROCESS workflow
    - Deploying the data download portal using the PROCESS platform (e.g. to immediately distribute the results of the computation).

As the primary purpose of UC#3 is to evaluate the ease-of-use of the extreme data solutions in contexts that serve emerging user communities, we assume that no quantitative metrics will emerge from UC#3 that wouldn't be covered by the other use cases. Rather, the success will be determined by mapping the new services to be integrated in a way that they serve the needs (both explicit and unarticulated) of the stakeholder groups observed or engaged with in the activity streams 1-3. Since at the moment the existing UC#3 requirements can easily be met with the PROCESS architecture, we can consider the approach being validated as far as UC#3 is concerned.

4.1 UC#3 Scenario

The basic scenario considered in the activity stream #1 (downloading the data set or parts of it and processing them on infrastructure that is) is primarily dependent on the level of general interest in the published datasets. The current dataset shows a steady but relatively low level of interest. The impact of the cross-linking of the UNISDR dataset with the recently published Climex data will be studied in the coming months [Jan, updated statistics?]. The project is also in discussions with the active disaster risk-related data producers, such as ongoing LEXIS project.

The existing link with the DMCC+ activity (supported by EOSC-Hub) has been activated for a feasibility/design study that would leverage existing data sets, tools and solutions in a pilot project aiming at extending the analysis of direct disaster impact towards policies and strategies for risk mitigation and adaptation. While in early stages, the initial results indicate that this approach could be used to validate PROCESS approach by stakeholders that are involved in operational civil protection activities. This input would allow more precise prioritisation of PROCESS features to integrate in the UC#3 service.

5 UC#4: Ancillary pricing for airline revenue management

The developments of UC4 were separated into 3 main directions:

1. Model training stream
2. Query service development
3. Data acquisition and extraction

The model training stream is targeting the creation of a Docker container where the mathematical model is trained with the acquired data. As stated in D4.3 the main requirement for the machine learning platform was to enable data scientists with less experience or weaker programing skills to efficiently create and evaluate different algorithms. The container provides the H2O.ai platform packaged together with R statistical computing environment enabling a wide variety of libraries and algorithms to use with the popular R language. As a first approach in UC4 we developed a probability estimation for booking an additional ancillary service with the flight booking. For the training data the generated ancillary data as described in D2.1 was used. The trained models are available as JSON model descriptors, which are downloadable after the training has finished. Since the targeted environment for the UC#4 Docker container is the UISAV data center with support for cloud resources, the container and the model training scripts were prepared to work with the HDFS file system provided by UISAV data center.

The query service development stream targeted to implement a REST service to score new ancillary booking test data requests. As stated in D4.1 the main requirement for this service is to provide very high throughput values together with a very low latency. The model with the parameter trained in the model training step is used inside as the scoring engine. The chosen VertX framework provides a non-blocking IO which enables the required high throughput and an asynchronous internal bus connected to a distributed cache solution aims for the low latency responses. The first measurements with the prototype showed promising

results, however further improvements towards clustering and load balancing with cache sharding is needed.

In the third stream the main goal is to acquire and use realistic data from a real customer database or data warehouse. Before any technical activities the main focus here was to negotiate an agreement with an airline to be able to use their data for our testing purposes. Even if there are multiple airlines interested in Lufthansa Systems development activities in PROCESS there are lot of concerns regarding data security and privacy. LSY has already data available for internal testing and data analytics, however there are further negotiations and agreements necessary until this data can be transferred into the PROCESS system landscape. Until further progress the main activity is to enhance the data generated by the already existing data generator and providing scripts to extract the data from the real data sources. The data generator was containerized into a Docker container and enhanced to save generated data as CSV files into an HDFS file system. The first tests with the data generator were carried out successfully in the UISAV environment.

The use case as an end-to-end solution will consist of a containerized pipeline of the data processing. It uses the Docker-based Cloudify variant of the PROCESS platform. We put the focus on using the LOBCDER stage-in and stage-out mechanism, although at this first stage the data stage is not yet working with the real data. Segmentation of the data is not yet done, however based on our first analysis of the real airline data there is a high chance that by segmenting the booking data set may lead to better performance and provides us with the possibility of parallel processing. In this case DataNet / Dispel could be used to provide the end user with the possibility of choosing between different market related data to start the workflow. There is also particular interest for the IEE (Integrated Execution Environment) with its support for the smooth and transparent access to the data required for the workflow processing.

5.1 UC#4 Scenario

The targeted scenario of the use case is a workflow triggered by the IEE. The model training container is initiated and deployed by the Cloudify orchestrator and data is made available by LOBCDER. The data will be generated by the Data Generator directly into the UISAV Hadoop environment, and this generated data is then referred to by LOBCDER.

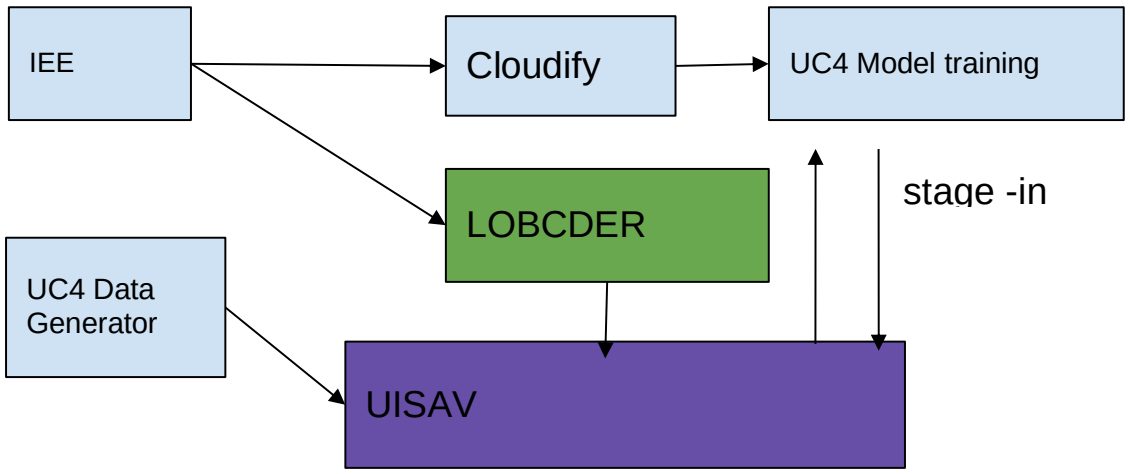

Figure 8: UC#4 workflow

5.2 UC#4 Benchmark

Currently our benchmarking is done with the generated data sets. Different sizes of data sets will be generated and the onboarding and processing time on those different sized data is measured. At a later stage processing time of different containers with different training algorithms will be compared with each other to help the end user find the best fitting mathematical model

The UC#4 will need to allow executing PROCESS components in a highly standardised, corporate Cloud environment in a manner that minimises new software components that need to be supported and monitored for compliance purposes. The work so far has shown that this seems possible to a large degree, and any additional OPEX costs will seem trivial in light of the new business opportunities solutions based on the UC#4 work represent.

6 UC#5: Agricultural analysis based on Copernicus data

In Use Case 5 the development was focused on the full integration of the SME's application PROMET, an established proprietary software solution that represents a large part of the key IPR of a German SME, into the PROCESS ecosystem. In summary, three main parts were enhanced:

- Deployment and integration of the UC owned software through an API container
- Integration of this UC-API into the PROCESS configuration and submission system
- Integration of a visualisation tool for output analysis

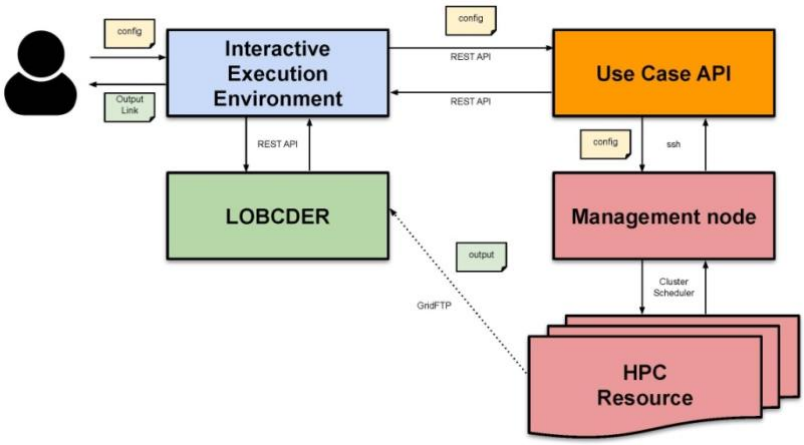

Figure 9: Overview of the API-Deployment for UC#5

The workflow was defined with the objective that the source code or other IPR of the SME cannot be accessed by anyone else than the use case owners themselves. This objective is met through a REST-based API. The configuration of the PROMET execution is configured in the PROCESS portal (IEE) and then transferred through an API container to the software. An overview of the API-Deployment can be found in Figure 9. This container was built as a template container to be adaptable to other SMEs and applications where the implementation details of the software need to be protected.

The actual execution is limited to one computing location, in this case the LRZ in Garching/Munich. Its management node allows to run a PROCESS workflow wrapper, which controls and deploys all necessary steps:

- Translation of the user configuration into applications specific configuration files
- Data Stage-In (currently only locally, but an integration for an external source via LOBCDER is in place)

- Deployment of the pre-processing application
- Deployment of PROMET
- Post-processing of the output files and visualisation depending on the user's configuration
- Movement of the actual output files via LOBCDER to PROCESS storage for user download

The complete sequence diagram of the interaction between the IEE, the UC API, the Centre's management node, the compute cluster and the data service is displayed in the Appendix in**Error! Reference source not found.**. Besides configuring the application and deployment the sequence diagram also shows how status updates can be pulled from the API so users can track the execution of the configured workflow. At the moment all displayed components can interact with each other and exchange data based on the specific protocols. On the use case owner site the application workflow is fully integrated, while the Data Service connection is in the process of implementation and will be ready in month 24, when the full pipeline starting from the end-user at the portal can be executed.

6.1 UC#5 Scenario:

Since this use case uses only the mechanisms from LOBCDER for Data Stage-Out and at a later time also for optional Data Stage-In, the validation and scalability possibilities within PROCESS are limited to the file transfer from the local computing cluster to a PROCESS storage resource. However, besides performance measurements this use case can show an efficient integration of software assets owned by European SMEs by providing end-user configuration and automatic deployment mechanism via the PROCESS ecosystem.

7 Conclusion

In terms of efficient data transfers across the PROCESS data centres, the preliminary test with the UC#1 which does not involve exa-scale data at this moment shows the potential of setup data transfer nodes. At this moment of the writing of the deliverable the tests with the DTN are not completed with the LOFAR scenario, and we expect to get more evidence of the necessity of the DTN with the LOFAR data, at this moment the LOFAR experiments can be regarded as the continuation of the initial evaluations presented in D4.3, the preliminary results are showing a certain overhead that is now being investigated further by reusing the benchmarks run on a production environment provided by SURFsara in Amsterdam.

7.1 Future work

In the coming months, we will continue testing scenarios described in D8.1 to validate reusability of certain services across the different use case. We will also engage with the Network providers in Germany, The Netherlands, Poland, and Slovakia to discuss the details related to setting up a network of Data transfer Nodes across the PROCESS Data Centre.

8 Appendices

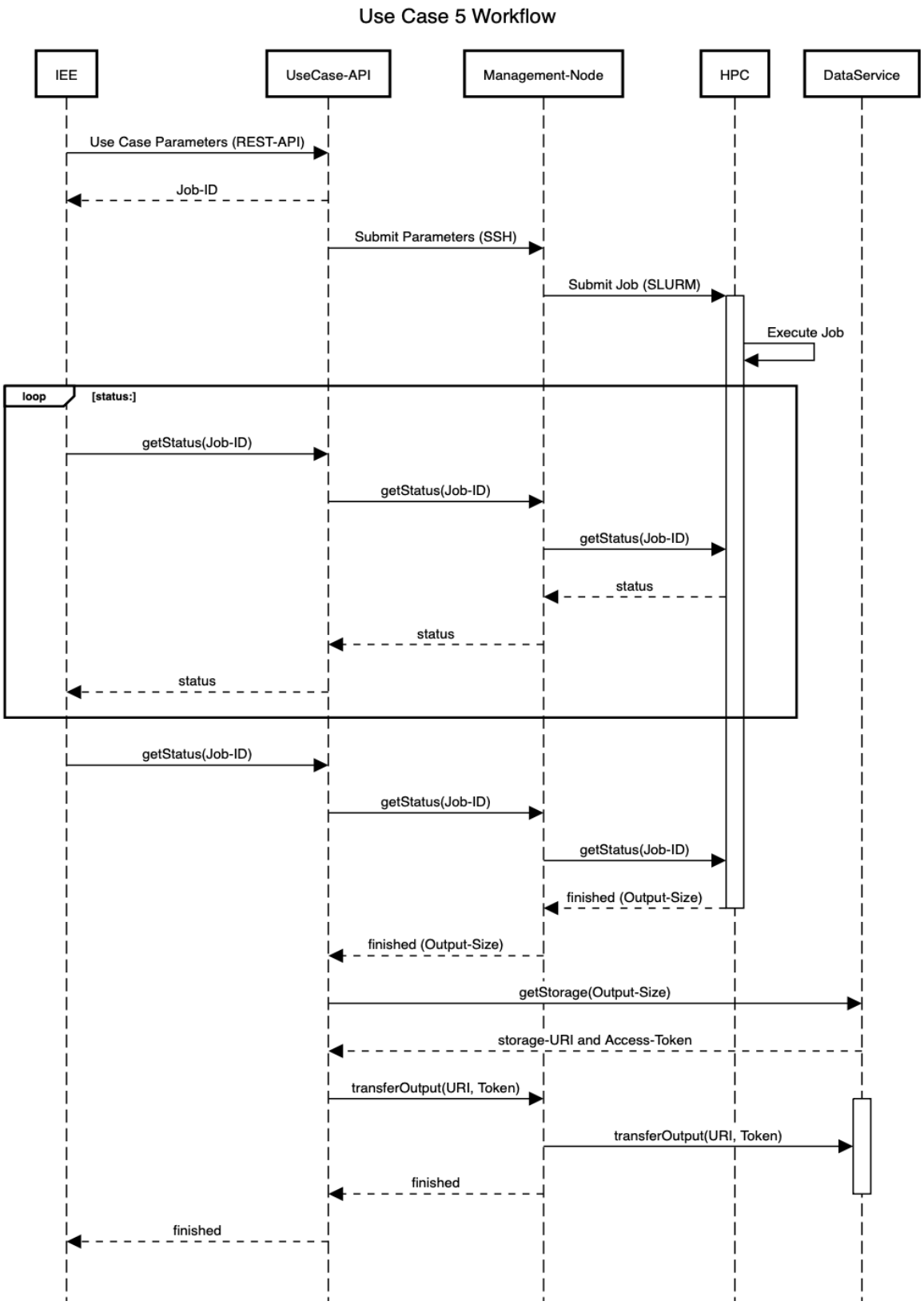

Figure 10: The complete sequence diagram of the interaction between the IEE, the UC API, the Centre's management node, the compute cluster and the data service of UC#5