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

D4.5	**Validation of the second prototype and final PROCESS architecture**		
Project:	PROCESS H2020 – 777533	**Start / Duration:**	01 November 2017 36 Months
Dissemination[1]:	Public	**Nature[2]:**	**R**
Due Date:	31.1.2020	**Work Package:**	**WP 4**
Filename[3]	PROCESS_D4.5_FinalPROCESSArchitecture_v1.0.docx		

ABSTRACT

This deliverable communicates the process and the results of the validation of the second prototype of the PROCESS platform and tools, as well as the final architecture of the PROCESS project. It is divided into two main parts. The first part is dedicated to the validation of the second PROCESS platform prototype (which is described in deliverable D6.2). The second part of the document presents the final PROCESS architecture which is the successor of the updated PROCESS architecture from deliverable D4.3. The architecture is extended by the Data Transfer Node (DTN) approach, has optimized resource management, and integrates European Open Science Cloud into the PROCESS platform.

[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] e.g. 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]	Hluchý, L.	UISAV	Deliverable coordinator
Contributing Authors[5]	Bobák, M., Tran, V.	UISAV	Writers
	Madougou,S., Maassen, J.,	NLESC	Writers
	Graziani, M., Müller, H.	HES-SO	Writers
	Somoskői, B.	LSY	Writers
	Schmidt, J.	LMU	Writers
	Cushing, R.	UVA	Writers
	Meizner, J., Nowakowski, P.	AGH	Writers
Reviewer(s)[6]	Belloum, A.	UVA	Reviewer
	Höb, M.	LMU	Reviewer
	Habala, O.	UISAV	Reviewer
Final review and approval	Höb, M.	LMU	Coordinator

Document History

Release	**Date**	**Reasons for Change**	**Status**[7]	**Distribution**
0.1	15.11.2019	The initial structure of the document	Draft	Consortium
0.2	22.11.2019	The final structure of the document	Draft	Consortium
0.3	6.12.2019	Initial text for all sections	Draft	Consortium
0.4	16.1.2020	Second update of all sections	Draft	Consortium
0.5	21.1.2020	Finalization of Validation of the second prototype	Draft	Consortium
0.6	22.1.2020	Finalization of Final PROCESS architecture	Draft	Consortium
0.7	23.1.2020	Review of the draft version	In Review	Consortium
0.8	27.1.2020	Preparation of the final version	In Review	Consortium
0.9	28.1.2020	Language correction	In Review	Consortium
1.0	31.1.2020	The final version of the deliverable	Released	Public

---

[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

This deliverable describes the validation process of the second prototype of the PROCESS platform and the final architecture of the PROCESS platform. The initial architecture is described in deliverable D4.1 and the first prototype of the PROCESS platform is validated in deliverable D4.3 which also contains the updated PROCESS architecture.

After 24 months, the PROCESS consortium delivered the second prototype of the PROCESS platform and used it to run the pilot application derived from UC2, which faced the challenge of insufficiently fast data transfer when deployed on the PROCESS platform. The validation described in D4.5 goes one step further as it focuses on integrating computing and storage resources across multi-site HPC as well as cloud centers.

To address the challenges faced in UC2, the architecture of the PROCESS platform presented in deliverable D4.3 has been updated as described in Section 2. The resulting PROCESS architecture will be used in the further implementation steps of the PROCESS platform.

List of Figures

List of Tables

Introduction

Deliverable D4.5 is the last in the series of architecture updates. The process was initiated in deliverable D4.1. The core requirements of the PROCESS use cases remain unchanged. The architecture is still driven by the service-oriented approach to exa-scale data processing applied to cloud and high-performance resources combined with advanced virtualization.

The main challenge of the first prototype was the processing of extreme-large datasets. We adjusted the architecture of the PROCESS platform to address this limitation by introducing the concept of nodes dedicated to transferring of data. The nodes are configured to perform high-speed data transfers over a wide area network (details are described in Section 2.2). The next significant architecture update is the optimization of service orchestration. The proposed change is focused on the improvement of cloud resource management. Last but not least, the architecture of the PROCESS platform is extended to the European Open Science Cloud (EOSC)[8] and the LOFAR portal.

To validate the second PROCESS prototype, we first went through the list of all use cases requirements to check whether or not they are supported by the second prototype. The second point was to assess the scalability of the system by studying and modelling different sources of overhead incurred by using the platform. These sources include various waiting/idling within PROCESS components, but also any extra scheduling different from the workload management systems (WMS) of clusters. The models are then used to assess PROCESS scalability.

The work documented in this deliverable is a result of common work across several work packages. WP4 is responsible for the design of the PROCESS architecture, but the scope of the deliverable is wider. According to this, the main goals were (with the responsible work package):

- Validation of the second prototype (WP8)
- Finale PROCESS architecture (WP4)

1 Validation of the second prototype

1.1 Validation approach

The main goal of this section is to describe the validation process followed to ensure that PROCESS solutions meet the requirements of the use cases (UCs) and user communities. It is achieved by deploying and running the UCs on the PROCESS platform. This will help us to assess the technology readiness level (TRL) of each component composing the PROCESS platform and TRL of the platform as a whole.

As defined in D4.3, the validation process follows a two-step approach where the functional requirements of the individual UCs are first assessed and then the pivotal non-functional requirement for PROCESS, which is platform operational overhead, is measured and analysed to assess the scalability of our solutions.

As described in deliverable D4.3, most PROCESS components/services running in the first platform prototype were integrated and met most of the UC requirements. Furthermore, the platform as a whole has reached TRL 5. In deliverable D4.5, we re-assess unmet UC requirements in the light of component/service evolution and re-evaluate the platform performance of the second prototype.

---

[8] https://ec.europa.eu/research/openscience/index.cfm?pg=open-science-cloud

1.2 Validation of UC functional requirements

In this section, we summarise UC requirements and their status in the first prototype and the change (if any) in the second prototype. UC requirements have been categorized in deliverable D4.3 in terms of PROCESS components/services, namely, extreme large-scale data/compute services, service orchestration and user interface. For the sake of simplicity, we enumerate the unfulfilled requirements without that categorization.

1.2.1 Use case 1

Table 1 summarizes the requirement analysis as described in D4.3 pages 8-9. Additional requirements that were not formulated in D4.3 are reported as “New requirement” in the third column of the table.

Table 1: Use case 1 requirement status.

Requirement	**Motivation**	**Fulfilled in D4.3**	**Fulfilled in the final prototype**
Support of technologies in Table 2 of D3.4 page 8	Toolboxes and libraries necessary for development	Yes	Yes
Support of new releases of Tensorflow and Keras	Such libraries are constantly subject to updates and bug fixes. The latest releases generally correct malfunctions of old releases	New requirement	Yes, through deployment of updated Singularity containers
Data transfer via SCP	Hospitals and private institutions may not guarantee open FTP access	New requirement	Yes. Site-to-site staging times are in D8.1 page 11
Support of distributed libraries for parallel GPU training: OpenMPI and Horovod	Horovod tool is a distributed training framework that supports Tensorflow and Keras	New requirement	Not met yet
Hardware requirements as of D4.1	No change needed	Yes	Yes
Security and privacy requirements as of D4.1	No change needed	Yes	Yes

In the following table, we report which requirements have been fulfilled in the second prototype and we give details on their validation and results:

Table 2: Additional remarks on the status of the additional requirements.

Requirement	**Status**	**Remarks**
Support for Tensorflow and Keras and new releases	Met	Docker and Singularity containers can be updated with the latest release of the software packages and then deployed in the PROCESS project.
Data transfer through SCP	Met	Testing site-to-site staging results are reported in D8.1 page 11. The SCP protocol is one of the slowest techniques, however, it guarantees that hospitals and research infrastructures not allowing FTP protocol could have an option to move data. As the second strategy, Data Transfer Nodes (DTN) have been tested, showing that DTN can speed up transfers. The use of DTNs could potentially accelerate transfer between sites even further.
Parallel execution of SLURM batches	Met	Three scenarios of deployment have been tested for UC1, namely: run of a single container on one computing site, run of several containers on a single computing site, run of several containers on several computing sites. The evaluation of performances in D8.1 page 9 highlights efficient data handling at AGH, and high-performance GPUs at UVA.

From these results and the data site-to-site staging tests (D8.1 page 11), it emerges that the ideal configuration for UC1 would be running the different use case layers at different computing sites.

The prototype testing showed that Layer I (see the workflow in Figure 1 and the description in D8.1 Table 1 page 8) profits best of the efficient data handling when running at AGH. The latest code refactoring shows the scalability of the Data pre-processing and the patch extraction pipeline (Figure 1) by distributing the tasks via the Slurm batch system. Our working examples show that linear speed-up can be achieved. Four different types of parallelization were implemented and released[9], which are described in Table 3.

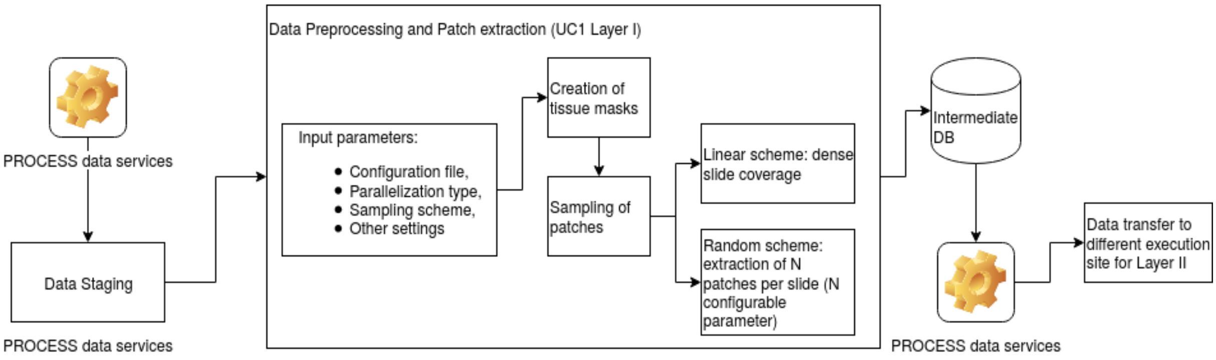

Figure 1: Data Pre-processing and Patch Extraction pipeline integration with PROCESS services (software Layer I workflow).

---

[9] https://github.com/medgift/PROCESS_UC1

Table 3: Description of the parallelization strategies adopted and achieved goals[10]*.*

Parallelization strategy	**Description**	**Goal**
Parallel patch extraction by randomized seed	Slurm job array where each task is configured with a different random generator seed.	Exploit randomness in the WSI point distribution in order to extract a different patch set at each batch run.
Parallel slide processing	Slurm job array where each task is assigned a different WSI (a single "patient case") while the random seed is the same	Under the hypothesis of infinite resources (i.e., available CPUs), with this script, we can achieve linear speed-up.
Parallel slide processing with several random generator seeds	The Slurm job array combines the two previous techniques: each task is assigned a different WSI (a single "patient case") while the random generator seed is varied over M parallel subtasks.	Under the hypothesis of infinite resources (i.e., available CPUs), this script achieves linear speed-up of <NUMBER_OF_SLIDES> x <NUMBER_OF_SEEDS>.
Parallel slide processing with several linear sampling windows	Slurm job array where each task is assigned a different WSI (a single "patient case") while a window (range) of mask indices is varied over M parallel subtasks.	This script analyses the slide in its integrity. No parts are left uncovered (dense coverage of the information). Moreover, under the hypothesis of infinite resources (i.e., available CPUs), this script achieves linear speed-up of <NUMBER_OF_SLIDES> x <NUMBER_OF_WINDOWS>.

Layer II, as described in D8.1 Table 1 page 8, is best run with distributed dense libraries. Implementation of parallel network training with the support of the distributed training library Horovod, Tensorflow and Keras showed at least 20% time save up with 2GPUs NVIDIA K80 (see Table 5 in D8.1 page 10). For this reason, the new requirement of “the support of distributed libraries for parallel GPU training, namely OpenMPI and Horovod” is set for future development. Particularly on the HPC clusters, this requires the integration of OpenMPI and Horovod with Singularity containers, on which we are focusing our current development efforts.

---

[10] For the experimental results see D8.1 pages 6-10 (Tables 2, 4 and Figure 1).

1.2.2 Use case 2

There were unmet requirements from the first PROCESS prototype as described in D4.3. In the following bulleted list, we analyse their status relatively to the second prototype.

- **UI requirements:** As most of the required functionality is provided by the Web interface for LOFAR[11] developed within the European EOSC Hub project[12], this product has been re-used and extended to include mechanisms for defining and executing pipelines. These mechanisms use the common workflow language (CWL) formalism and rely on open-source and publicly available NLeSC-developed tools, are Xenon-flow and Xenon[13], for the enactment and control of the pipelines. Work is being done to let IEE and LOFAR EOSC interface interact and the pipelines run through IEE.
- **Support for data staging and transfer to HPC sites:** As reported in D4.3, the staging was done manually through the traditional approach provided by the LOFAR long term archive (LTA). With the second prototype, LOBCDER now provides both a staging service and a transfer service using the storage resource management (SRM) protocol. Both services are provided not only with execution endpoints but also with endpoints for checking the status of the respective operations, which proved very handy during their integration into UC2. The services have been tested on DAS5 cluster in Amsterdam and are working consistently.
- **GridFTP (or alternatives such as FTS) support and DTNs:** As detailed in the previous point, provided data services use the SRM protocol, which has been demonstrated to be only slightly less performant than GridFTP[14]. As a consequence, to improve the data transfer performance, which is currently suboptimal as described in D8.1, we heavily rely on the implementation of the DTN infrastructure or equally well performing alternatives.
- **Horizontal scaling via multisite execution:** The ability to horizontally scale to a significant number of computing resources in order to run in parallel is paramount for UC2. Although most of UC2 pipeline steps actually run on a single node, these steps can occasionally use all the computing resources (cores) of a given node as the steps by themselves actually represent workflows or pipelines with parallel (sub)steps. The most compute-intensive steps which concern direction-dependent calibration can run distributed, but none of the tried tools direction-dependent facet (DDF) and FACTOR has proven production-readiness. Nonetheless, this requirement is still important as it will allow processing several observations in parallel. This can be done, for instance, by partitioning a large cluster such as SuperMUC-NG into sub-clusters and run a reduction pipeline on each one of them. Of course, for this scheme to work, all required services (notably, data (LOBCDER) and data post-processing (DISPEL)) need to be working on all clusters, which is not yet in place.

---

[11] https://github.com/EOSC-LOFAR

[12] https://eoscpilot.eu

[13] https://github.com/xenon-middleware

[14] *Optimizing large data transfers over 100Gps wide area networks*, Rajendran et al., CCGrid 2013, Delft, Netherlands

The table below summarizes the current status of the requirements of UC2.

Table 4: Use case 2 requirement status.

Requirement	*Status*	*Comments*
UI requirements	Done through EOSC 2	To be linked to IEE and integrated
Data staging and transfer	Implemented in the second prototype	Optimization needed
DTN infra	One DTN at UvA site	Testing needs more DTNs
Multisite execution	Work in progress	This is really needed.

We note that while the first two requirements have been fulfilled, the last one is still under development. As of the DTN implementation, it needs to be tested on this use case.

1.2.3 Use case 4

UC4 has evolved a lot since the last release of the PROCESS platform, new requirements have been formulated for UC4 and described in D4.3. Table 5 summarizes those requirements and their fulfilments.

Table 5: Use case 4 requirement status.

Requirement	*Status*
HDFS support	fulfilled
Switch from HPC to Cloud and use of Docker should be done by service orchestration (Cloudify)	fulfilled
Integration of IEE and Cloudify	fulfilled
Compliance with GDPR	Out of scope of UC4

HDFS support: The requirement of UC4 demands HDFS/Hadoop for data storage purposes. It is now satisfied through the Platform Cloud Services by providing HDFS resources. The model training application container connects and reads data from this storage. Since the availability of HDFS support, the application was also extended to benefit from this feature.

Switch from HPC to Cloud and use of Docker should be done by service orchestration (Cloudify): With this requirement, we wanted to ensure better portability and exploit cloud resources even outside of PROCESS platform. The Docker containers for the UC4 application were created and the corresponding Cloudify Blueprint resources implemented.

Integration of IEE and Cloudify: We formulated this requirement to provide the UC4 application users with high transparency of the underlying infrastructure. The IEE and Cloudify were technically integrated. The above mentioned Cloudify Blueprints are selected, deployed and executed automatically when the user triggers it from the IEE graphical user interface.

Compliance with GDPR: After further analysis of the data sources, we identified that the original data is already anonymized, since this is already in the GDPR scope. So we don't need further actions to take, and this requirement is not valid any longer.

1.2.4 Use case 5

As stated in previous documents, Use Case 5 presented a unique requirement due to its IPR issues. The integration of such use cases has been described in D4.3 and elaborated in D2.2. To meet the requirement of this and similar use cases, an API was developed that acts as a proxy for job submission and data acquisition. This API was integrated with the PROCESS architecture and instantiated for Use Case 5. The previously specified functional requirements have been met and the integration of the API has shown no overhead. The API is fully integrated with the IEE supporting a configuration of execution parameters of the use case, job submission, monitoring and stage-out of results for the end user.

1.3 Platform performance analysis

In D3.2, we modelled the behaviour of the platform overhead. We proposed to use a correlation analysis to get insight into the data and compensate for the lack of sufficient data. Figure 2 shows the overhead given by the model for both IEE and UC2 environments[15]:

- For IEE the overhead which is expressed as a linear regression of the data shows a very slow variation of the overhead in the function of the number of containers, which is very important for PROCESS scalability (see Figure 2-a (left)). The small value of the coefficient R_2 measuring the goodness-of-fit is the result of the lack of sufficient data in IEE. According to this model, the overhead of processing of the entire LOFAR LTA archive (around 1800 observations of 16TB) would be only about 7s. However, we still need to confirm the model prediction by using more data for the next validation in deliverables D3.3 and D8.2.
- For the UC2 the overhead, shown in Figure 2-b (right), is higher compared to the overhead obtained for the IEE, namely 4683s (~1.3h). In contrary to IEE, here, the model exhibits a more acceptable R_2 value (0.88) which give us more confidence in the value predicted by the model. We note here that we had enough data to perform our experiments. In the final setting, we expect that the overhead of the UC2 environment will be in the order of that of the IEE as the latter is the final environment of all UCs. Besides, 1.3 hours is negligible compared to the months it would take to process all observations in the archive, each one taking four to five days to process (<0.0007%).

---

[15] IEE and UC2 environments are described in Section 4, page 16, in deliverable D3.2: Application of the Prediction Model to actual Measurement Results and Conclusion.

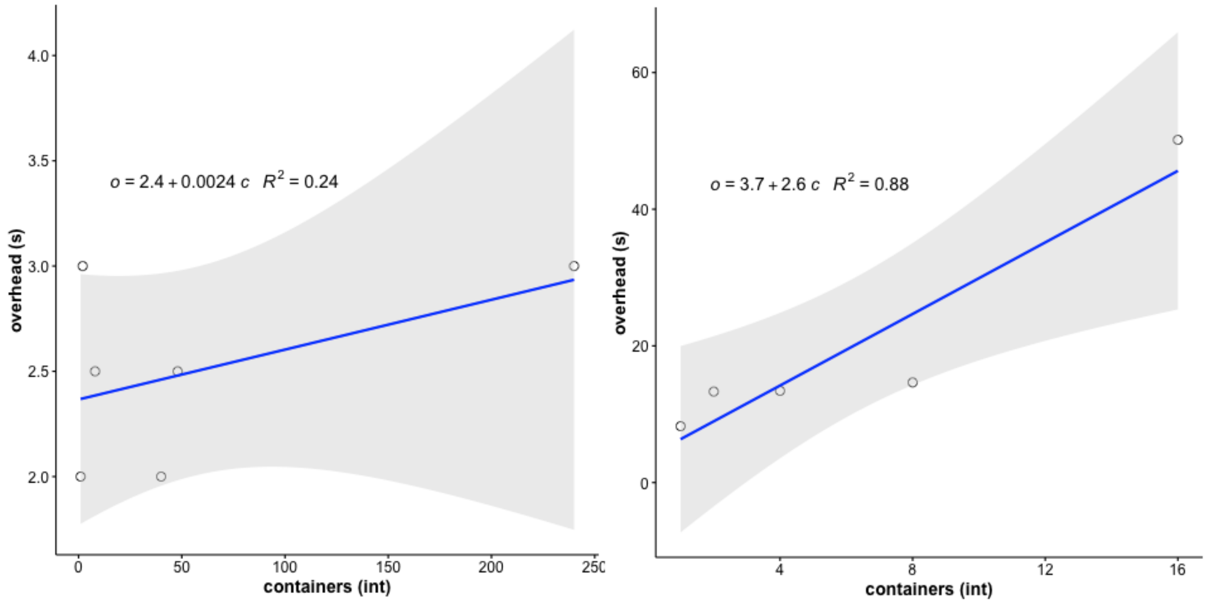

Figure 2: PROCESS overhead models.

Similar to the general overhead, we also modelled the behaviour of the platform's scheduling overhead. We used the best practise to exclude overhead due to WMS, however, it has an impact on the queueing delays reported in D3.2 and modelled here. As illustrated in Figure 3, for both IEE (left) and the UC2 environment (right), the value of the goodness-of-fit is pretty high (>= 0.90) bringing trust to the corresponding models. Again, the IEE model shows a slow variation of the scheduling overhead proportionally to the number of containers ($o = 0.044 * c + 1.9$) whereas for UC2, the variations of both dimensions are of the same order ($o = c + 1.4$). Following those two models and their prediction results, the PROCESS framework introduces a small and acceptable overhead in terms of scheduling, which is not considered as any obstacle. However, this needs to be confirmed in deliverable D3.3 by validating the models with more data.

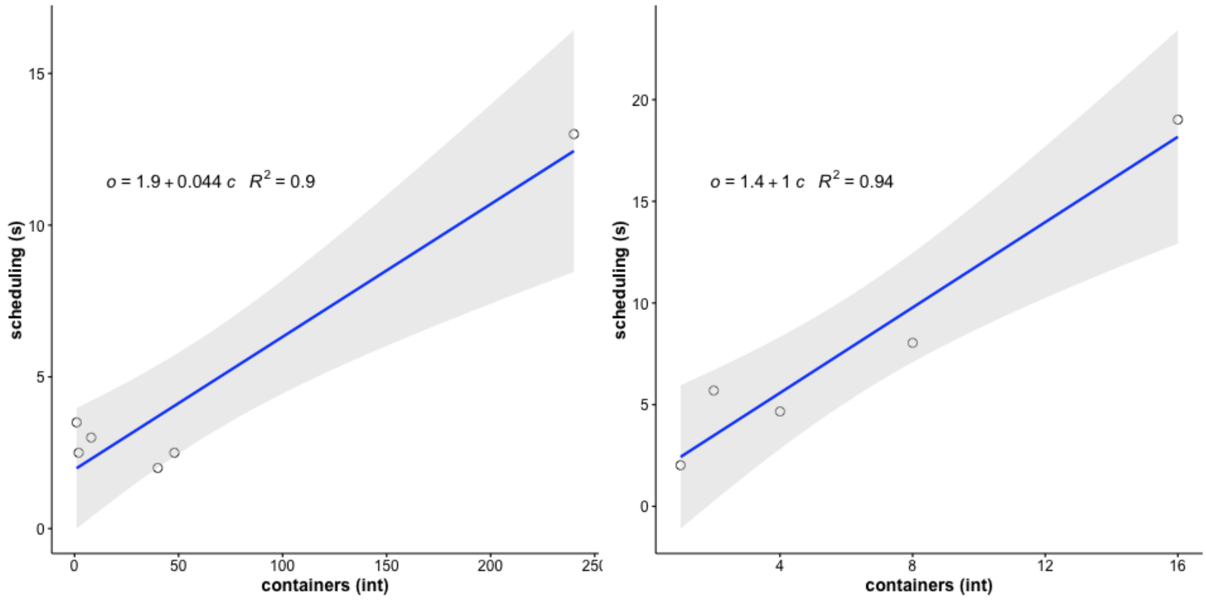

Figure 3: PROCESS scheduling overhead models in IEE (left) and UC2 (right).

We modelled and summarized data staging and transfer measurements done in deliverables D3.2 and D8.1. In D8.1, section 3.2, pages 13-14, we showed the data staging times in function of the size at two LOFAR LTA locations, Amsterdam, NL and Poznan, PL. In D3.2, we modelled the average behaviour of the staging on these two sites as illustrated in Figure 4.

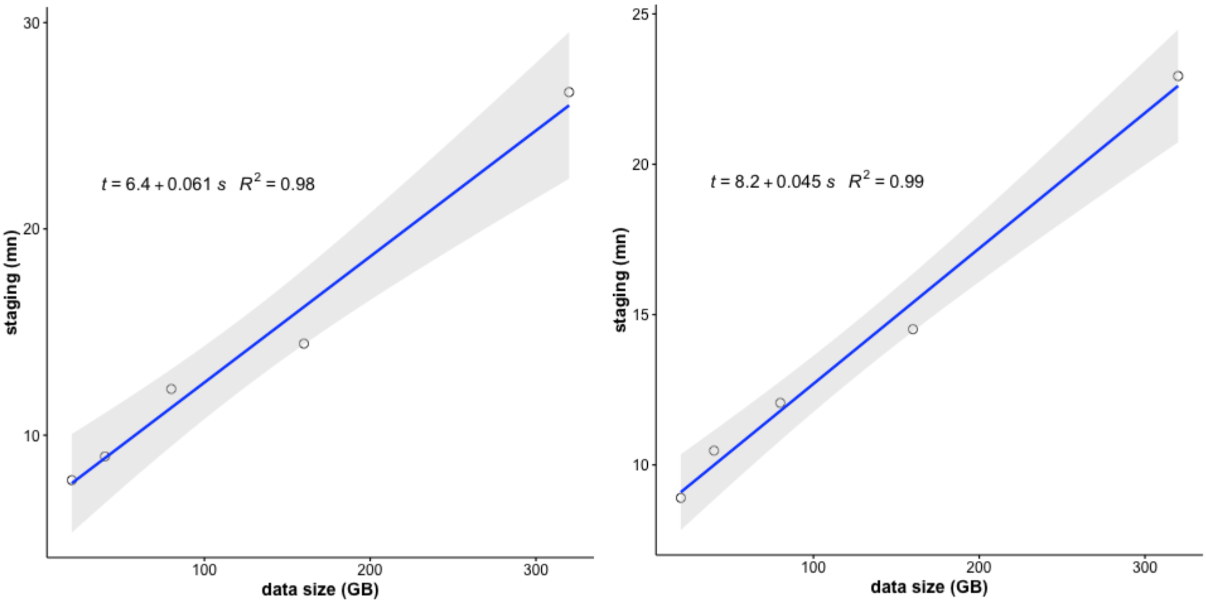

Figure 4: PROCESS data staging time models in Poznan (left) and in Poznan and Amsterdam (right). The staging time (t) in minutes) is expressed as a linear function of data size (s) in GB.

According to the average model, it would take 745,480 minutes (half a day) to stage a LOFAR observation of 16TB.

Transfer speed measurements between LOFAR LTA locations and PROCESS computing sites on one hand (D8.1), and between the computing sites on the other (D3.2), have also been conducted. The first experiment did not use DTNs showing a poor average transfer performance (5-12 MB/s), whereas the second approach with DTNs shows promising transfer rates of up to 550MB/s. The latter experimented only on UC1 needs to be extended to the other use cases, for instance UC2 requiring high-speed data transfers from LTA locations to the computing sites.

1.4 Validation conclusion

To validate the second PROCESS platform prototype, we first compared the UC requirements against the new features of the platform, and second, we evaluated the performance of the platform as a whole, mainly measuring the overhead incurred by a usage of the current implementation of the middleware as more and more services are being added to it.

For the first part, although most of the not supported requirements in the first prototype are now fulfilled, some are still work in progress while others have changed into new requirements. For instance, the requirement "support for distributed libraries for parallel GPU training: OpenMPI and Horovod" of UC1 or that of "horizontal scaling through multisite execution for UC2" are not implemented yet.

For the second part, as in deliverable D3.2, we measured and modelled different metrics defined in D3.1, focusing on a platform overhead and a scheduling overhead. The analysis of the models shows that the PROCESS platform introduces a low overhead, thus validating the architecture and implementation choices made for PROCESS. As of the scheduling within the

PROCESS platform, it creates a moderate overhead on the system but does not constitute a showstopper. One catch with all the above models is that they are drawn from relatively small datasets, we hope to overcome this issue in the final prototype. Given that the PROCESS platform has a very low overhead, it can then scale to a large number of clusters capable of dealing with exa-scale datasets.

Finally, in terms of technology readiness level (TRL), we showed that the platform in its first prototype already reaches TRL 5 by validating its components and their interactions. Now, the second prototype is nothing but the first prototype with more featured components and even better interactions, shown to be working with real world applications (UCs) on realistic environments (the PROCESS infrastructure). Consequently, the PROCESS second prototype reaches TRL 6 as planned.

2 Final PROCESS architecture

2.1 Integration of the LOFAR portal

The LOFAR community is driving the development of UC2. The Netherlands eScience center extended the actual LOFAR portal towards a usage in the EOSC context. Therefore, besides the selection of an observation also a submission system was integrated.

This submission system connects via an API also to the PROCESS IEE, the central part of all deployments to Cloud and HPC resources done by PROCESS. To enable this communication, the PROCESS architecture had to be extended by an IEE adapter to external sources (see Figure 6). Thereby, the IEE is able to list all available pipeline defined for the LOFAR computations and deploy the entire workflow. The complete sequence of actions is listed in Figure 5.

As it can be seen, an astronomer uses the LOFAR portal as the entry point, where an observation and configuration parameters are defined. Inside the portal, the computation is started and will trigger a submission of the workflow via IEE, which calls LOBCDER to stage in the data, deploy the container and makes the output available.

The addition of the API interface in the architecture was necessary, since the LOFAR community is used to the existing portal. Therefore, to not switch the user interface, the actual deployment is abstracted from the astronomers.

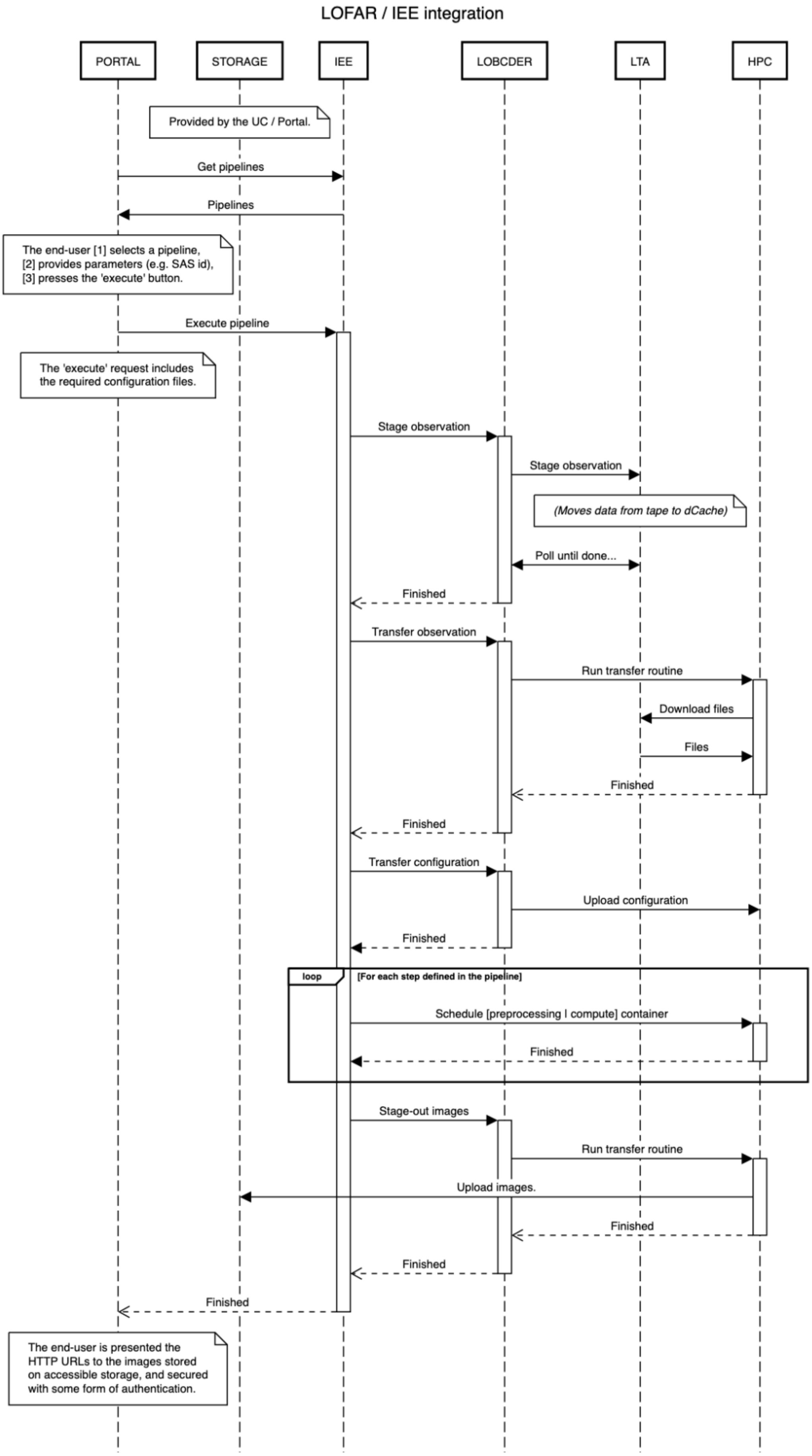

Figure 5: LOFAR portal deployment option integration into IEE.

2.2 Adaption of the PROCESS architecture

As the project progressed, new exa-scale aspects needed to be investigated more deeply. The improvements were focused on data transfer which was identified as the main bottleneck of the PROCESS platform prototype. The problem is addressed by a dedicated set of nodes - data transfer nodes. The second significant update is dedicated to the optimization of computing resources management. The second prototype of the PROCESS platform supports both cloud and HPC resources through dedicated managers Cloudify for cloud resources and Rimrock for HPC resources. Last but not least, Cloudify is successfully integrated with the European Open Science Cloud. The overall schema of the final PROCESS architecture is depicted in Figure 6.

Data Transfer Nodes: DTNs are special hardware nodes that are dedicated to the transfer of data. Such nodes have high network bandwidth and sizable storage that can be used as caches to transfer data at high speed between DTNs. Tuning of these nodes and their connections often requires network experts. In our architecture, we assume DTNs are available, pre-optimized and have some means to be programmed e.g. having the ability to deploy containers onto the DTN. LOBCDER's role is to integrate DTNs through this programmability and be able to copy data to/from DTNs.

Atmosphere replacement: Atmosphere was the initial solution considered in the PROCESS consortium to manage the orchestration of multi-cloud resources. However, as indicated in D4.4, further investigations showed that Cloudify is a more robust, more widely adopted by the cloud community, and present similar functionality, so in the interest of avoiding duplication of functionality, the PROCESS consortium decided to replace Atmosphere by Cloudify.

EOSC integration: Cloudify can deploy VMs on any compatible Openstack sites, including sites in EOSC-Hub Federated Cloud infrastructure[16]. It opens the door for users to access computation resources and integration with the EOSC. For the full execution of use cases on EOSC Federated Cloud, other components should be integrated with EOSC-Hub, too, mainly user portal and data infrastructure.

---

[16] EGI Compute Cloud https://www.eosc-hub.eu/services/EGI%20Cloud%20Compute

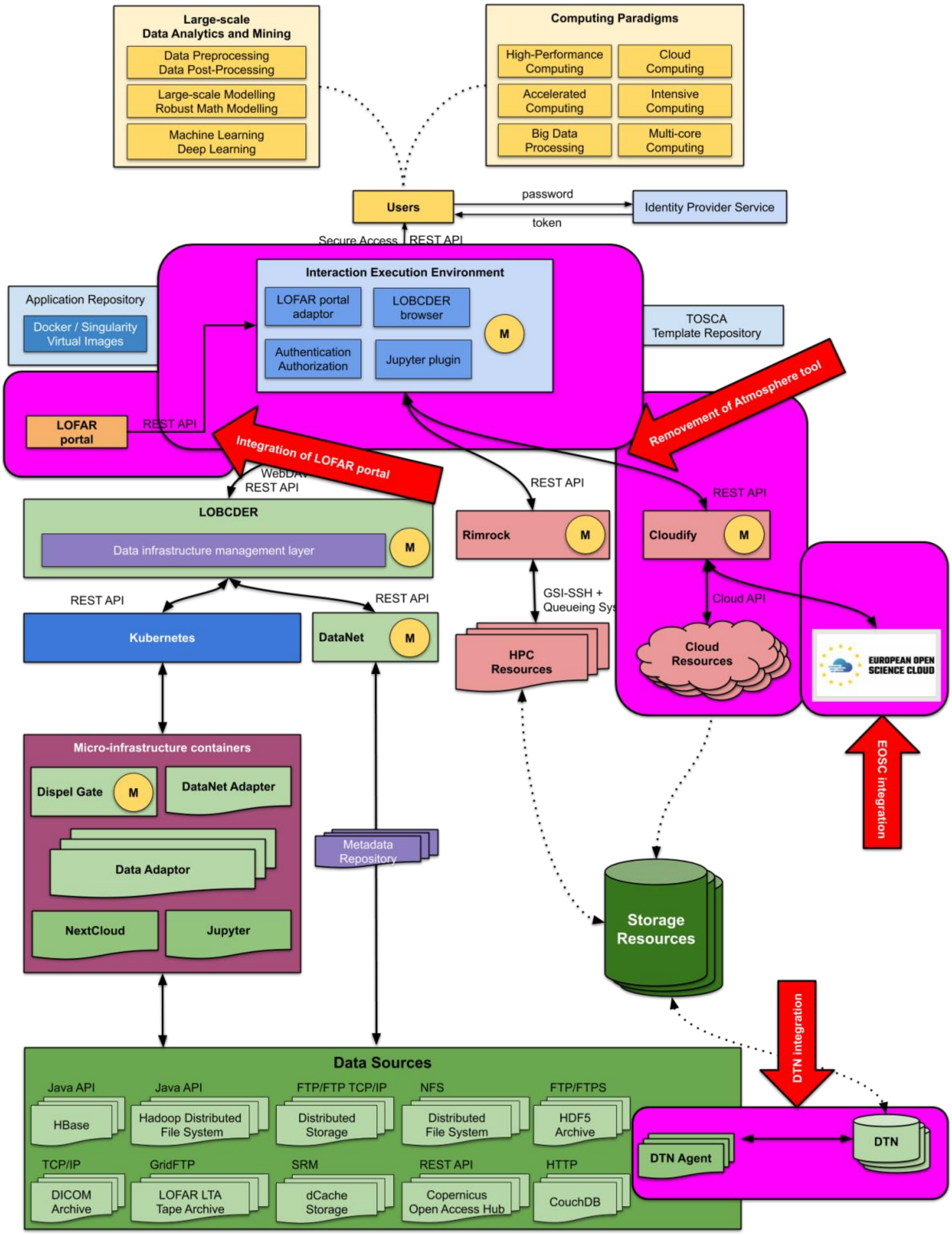

Figure 6: Final PROCESS architecture with new updates.

3 Reference exa-scale architecture

The last version of the reference exa-scale architecture addresses the requirements coming from the requirement analyses (see deliverables D4.1, and D4.3) as well as the validations of the platform prototype (see deliverables D4.3 and D4.5). The reference architecture design was initiated in deliverable D4.1 and updated in deliverable D4.3. The PROCESS consortium has not identified new functional requirements; thus, we consider the reference exa-scale architecture described in D4.3 as the final one.

4 Conclusion

The deliverable presents the final version of the PROCESS architecture and the validation of the second platform prototype. Recommendations obtained during the project were applied to the updated PROCESS architecture and led to its final version.

In this deliverable, we also present the validation of the second prototype of the PROCESS platform. Besides a couple of requirements which will be likely available in the next weeks, all use case requirements are fulfilled in this prototype whose performance in terms of overhead is analysed and shown to be light, paving the way towards a scalable system capable of processing exa-scale datasets. The deliverable also shows that the whole system now reaches TRL 6 as expected.