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

D3.1	Performance modelling and prediction		
Project:	PROCESS H2020 – 777533	Start / Duration:	01 November 2017 36 Months
Dissemination[1]:	PU	Nature[2]:	R
Due Date:	31 January 2019	Work Package:	WP 3
Filename[3]	PROCESS_D3.1_Performance_modelling_and_prediction _v1.0.docx		

ABSTRACT

This deliverable is the foundation for the PROCESS performance modelling and prediction approaches. With this foundation, it will become possible to steer the future design, development and validation efforts. A special challenge in the PROCESS project is the variety of environments the developed software is supposed to support, requiring an uniform, easy-to-use and straightforward performance model.
Hence, this deliverable first determines all relevant measurands for the different workflows. Heron, we present an initial approach to compose the components for predictively modelling a workflow's performance on the PROCESS architecture.

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	Organisation	Role / Title
Deliverable Leader[4]	Bubak, Marian	AGH	Coordinator
Contributing Authors[5]	Meizner, Jan	AGH	Writer
	Höb, Maximilian; Schmidt, Jan	LMU	Writers
	Maassen, Jason	NLESC	Writer
	Belloum, Adam	UvA	Writer
Reviewer(s)[6]	Guggemos, Tobias	LMU	Reviewer
	Nowakowski, Piotr	AGH	Reviewer
	Somoskoi, Balázs	LSY	Reviewer
Final review and approval	Höb, Maximilian	LMU	Reviewer

Document History

Release	Date	Reasons for Change	Status[7]	Distribution
0.0	01-10-2018	Structure	Draft	
0.1	01-12-2018	Measurands/Model Definition	Draft	
0.2	08-01-2019	Model Development	Draft	
0.4	28-01-2019	Review-Process	In Review	
1.0	31-01-2019	Release	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 4
List of Figures 5
List of Tables 5
1 Introduction 6
1.1 Performance modelling approaches 6
1.1.1 Overview and classification 6
1.1.2 PROCESS Performance Model 7
1.2 Structure of the Document 7
2 Identification of Measurands 8
3 Development of a balanced Prediction Model 12
3.1 Runtime Composition 12
3.2 Model Verification 13
3.2.1 Benchmark Application 13
3.2.2 Use Case Workflows 14
3.3 Conclusion 14
4 Measurements 14
5 Application of the Prediction Model to actual Measurement Results and Conclusion 14
6 References 15

Executive Summary

This document presents the foundations of the performance modelling and prediction approaches that the PROCESS project will use to steer its design, development and validation efforts. The broad range of environments the PROCESS software will run on presents obvious challenges in the development of a uniform, easy-to-use and straightforward performance model. The necessary streamlining and simplification of the approach should not omit any relevant aspects that are determining the actual performance as observed by a user.

As a way to balance these conflicting requirements, the project will use a solution based on measurable performance metrics, complemented by a mathematical model that allows extrapolating performance on systems that are considerably more complex than the existing ones. The extrapolation is necessary to understand the impact of advances in the capacities of individual components on the execution speed of complex workflows.

The model used by the project assumes that exascale applications can be modelled as pipelines consisting of the input data stage-in, processing and (result) data stage-out steps. However, for workflows composed of several dynamically configured and deployed components, the set of performance components need to be able to analyse the execution in a more fine-grained manner. The full set of metrics is detailed in Section 2.

It should be noted that some of these steps depend on user input, therefore, the overall execution time will depend on the expertise and skills of the user. There are also considerable differences between situations where all the necessary resources belong to a single system, on multiple platforms controlled by a single organisation or in a federated system crossing organisational and geographical boundaries.

To focus performance-related development efforts, the PROCESS performance model groups the metrics into the categories *Overhead*, *Data Transfer*, *Scheduling* and *Execution Time.*

In the performance model presented in Section 2, the overhead represents the times introduced by the use of the PROCESS software, while the data transfer and execution time components are primarily dependent on the performance of the networking, storage and computing hardware available. Scheduling is highly dependent on the number of competing jobs and the policies (e.g. priority queue available for the job). However, similar to the characteristics of the underlying hardware, scheduling is an issue that cannot be influenced by the design of the software.

The relative impact of these four categories on the system-level performance as experienced by the user can vary dramatically. Therefore, the ultimate goal of the project is to develop a user-configurable workflow that can be used to complement actual use case software in the evaluation of the PROCESS platform. However, it should be noted that the use cases already stress the different aspects of the equation in a quite comprehensive manner.

The conceptual model presented in this deliverable will be updated and further elaborated with the actual measurement data based on the use cases and the benchmark workflow being available. The updated performance modelling and prediction results will be the focus of deliverable D3.2.

List of Figures

Figure 1: Sequence diagram describing the steps involved in the execution of a typical application scenario 9
Figure 2: Three measurement scenarios: (a) Single container – single site, (b) Multiple containers – single site, (c) Multiple containers – multiple sites. In all three scenarios Stage-In and Stage-Out will limit the overall system performances unless the wide area network data transfer issue is addressed. 11

List of Tables

Table 1: Performance Modelling Approaches, cited from Performance Modelling, David Henty [PMO] 7
Table 2: Description of the PROCESS measurands 10

1 Introduction

This deliverable provides an initial approach to model the performance of the PROCESS infrastructure and its possible scalability towards exascale workflows. Based on D3.1 the upcoming deliverables D3.2 and D3.3 will enhance and complete the process of developing a performance model and provides the opportunity to present predictions of the architecture's behaviour towards extreme large workflow executions.

Given that no single high performance computing system currently available or contemplated is expected to provide exascale performance, we need, on the one hand, access to multiple computing centres capable of running federated computational workflows, while on the other hand, software which can be deployed and coordinated not only across several nodes, but across different locations in Europe, which we refer to as sites. For the use cases presented in deliverables D2.1, D4.1 and D5.2 and based on PROCESS architectural design decision, we consider containerisation a prerequisite. In order to technically facilitate this decision, we follow an approach which would permit us to containerize architectural elements as well as to push all use cases to implement their execution in containers. This will allow for deploying instances of independent executions on sub-sets of a given data set on different local nodes, and also on different geographically distributed sites.

However, the hardware and the software development towards exascale is subject to ongoing research and we have to face the challenge to predict a behaviour that cannot be verified within the lifetime of this project. Therefore, we need to develop a prediction model based on measurable performance indicators and use them to extrapolate runtime behaviour towards a much higher scale. The model needs to meet the requirements to predict the behaviour of all our services and the PROCESS infrastructure as a whole, but must be flexible to adapt to new requirements coming from future and new applications.

To distinguish the most common approaches for performance prediction models, we first provide an overview and classification of up-to-date performance modelling and prediction methods, on the basis of which we present the PROCESS approach to performance modelling.

1.1 Performance modelling approaches

Performance modelling is used in many computational and storage systems around Europe. Regarding the exascale challenge, also other EU projects also examine the needs and conclusions to enable exascale performance.

One representative is the CRESTA[8] project (Collaborative Research Into Exascale Systemware, Tools and Applications) proposing a framework on software and tool developments for scientific end-users. Their solution is limited to local site needs and deals mainly with hardware decisions which supercomputing centre operators will face over the next few years.

1.1.1 Overview and classification

David Henty from the Edinburgh Parallel Computing Centre (EPCC) is one of the CRESTA project partners. In his publications (e.g. Performance Modelling [PMO]) he gives an overview on generic performance modelling techniques, along with a corresponding classification. In Table 1 we summarise the four main performance techniques proposed by David Henty, starting with raw measurements, through benchmarking and simulations, all the way to complex analytical modelling with a large number of parameters.

[8] https://www.cresta-project.eu

Technique	Description	Purpose
Measurement	running full applications under various configurations	determine how well application performs
Microbenchmarking	measuring performance of primitive application components	provide insight into application performance
Simulation	running application or benchmark on software simulation	examine "what if" scenarios e.g. configuration changes
Analytical Modelling	devising parameterized, mathematical model that represents the performance of an application in terms of the performance of processors, nodes, and networks	rapidly predict the expected performance of an application on existing or hypothetical machines

***Table 1**: Performance Modelling Approaches, cited from Performance Modelling, David Henty [PMO]*

All the techniques mentioned in Table 1 may prove useful the PROCESS project:

Measurement
Measurement may involve simple observables as well as complex models. In Section 2, we will define meaningful measuring points in the execution sequence, whose values are to deliver input data for further modelling and prediction steps.

Microbenchmarking
Due to the inherent complexity of exascale use cases, ordinary sample applications are not sufficient to contribute to a fundamental performance model. Nonetheless, microbenchmarking will be used to identify performance bottlenecks in the PROCESS architecture and assist in debugging and verifying its correctness.

Simulation and Analytical Modelling
Executing and measuring a given application running on PROCESS in different configurations and settings comprises the input dataset for this step. The goal is to extrapolate the application's behaviour and operation with the observations. The resulting model will allow predicting runtime behaviour beyond the measured configuration scales, which gives us a chance to forecast the performance on the exascale level.

1.1.2 PROCESS Performance Model

Based on the previous description, we choose a measurement-based approach with extrapolation through analytical modelling. First the measurands are identified and measurements will be collected. Subsequently, a microbenchmark to evaluate these measurands will be developed. Finally, to predict the performance of PROCESS, we will use these results to create an analytical model that will enable us to extrapolate the performance based on the obtained measurements.

1.2 Structure of the Document

In Section 2, we identify the appropriate measurands within the PROCESS infrastructure and present their occurrence within the PROCESS execution workflow. Based on these eight measurands the prediction model is established in Section 3, which will later facilitate extrapolation of the PROCESS scalability. The Sections 4 and 5 will be fleshed out in the following deliverables, D3.2 and D3.3, with the measurement results for a test application along with actual use cases, and, finally, the model will be applied to actual measurement values to extrapolate the performance.

2 Identification of Measurands

In the previous section we categorized the different approaches to performance modelling and prediction. One of which was a measurement-based approach with extrapolation for performance prediction. To achieve this goal, it is necessary to identify the appropriate measurands within the PROCESS infrastructure that can be used to model the performance of the infrastructure and predict its scalability.

We would like to stress here that the hardware infrastructure, including computing, storage, and network resources, will have a significant impact on the performance of PROCESS services. However, as a project, we have no real influence on this part of the infrastructure. Therefore, our performance measurands will focus on the overhead introduced by the software services, while also measuring all other relevant values to identify relations between them.

In the absence of true exascale systems, our objective, as stated in Section 1, is to achieve exascale performance by combining the power of geographically distributed datacentres. Unfortunately, standard configurations of computing centres are typically optimized for internal transfers of data rather than for cross-site transfers. While technical solutions to optimize data transfers exist, such as the Data Transfer Nodes[9,10], implementing those solutions is beyond the scope of the project. In PROCESS will try to conceal the data transfers by overlapping data transfer with computing, or use pre-fetching and caching to minimize the need for synchronous data transfers.

Based on the five use cases defined in PROCESS, we can think of a typical application as a pipeline of data processes which typically requires a data stage-in step followed with an execution step, and finally a data stage-out step. The time required for stage-in and out is expected to be significant, due to the necessary transfer of data between datacentres.

Figure 1 presents a sequence diagram describing all steps involved in an application scenario execution. For each step, we define the time corresponding to its completion as follows:

T1: Configuration

The Interactive Execution Environment provides an end-user web portal, where each run of any application needs to be configured. For the different use cases, these configurations vary as shown in the deliverables D4.2 and D5.2.

T2: Deployment Strategy

Part of T2 is the time needed to decide on which computing site[s] and storage site the containers and their data will be deployed. This step also needs to initialize the required micro-architecture.

T3: Stage-In

This step involves access to data services hosted at the selected data centre. However, if PROCESS can make use of caching, proactive pre-fetching or pre-processing, the impact of T3 upon the overall execution performance can be substantially reduced.

T4: Container selection

The workflow that has been defined in T1 specifies a container that will be executed as well as the required version. This version needs to be fetched from a container repository and later deployed as a job in T5.

[9] Building User-friendly Data Transfer Nodes, https://www.delaat.net/posters/pdf/2018-11-12-DTN-SURFnet.pdf

[10] Pacific Research Plattform https://bozeman-fiona-workshop.ucsd.edu/materials/20180303-dart-dtn-strategic-asset-v1.pptx/view

T5: Scheduling

The time a job spends in the queue of the compute resource. This time can vary and will be hard to predict since it is affected by each compute site's scheduling system, which is not under the control of PROCESS infrastructure. We may, however, be able to establish an upper bound for the queue wait time, that could be included in runtime prediction.

T6: Execution time

T6 is the time a given job takes between from leaving the queue and completing its calculations on the compute resource. This time is determined by the performance and scalability of the application on the selected compute resource. To predict this time, an application specific performance model is required.

T7: Stage-Out Strategy

Once the job is complete, it may have generated large amounts of output data that needs to be transferred from the compute resources back to the PROCESS permanent storage infrastructure. Based on the amount of data and the specified workflow the data service needs to choose a suitable stage-out strategy.

T8: Stage-Out

With the appropriate stage-out strategy the output data now needs to be transferred to the selected storage resource.

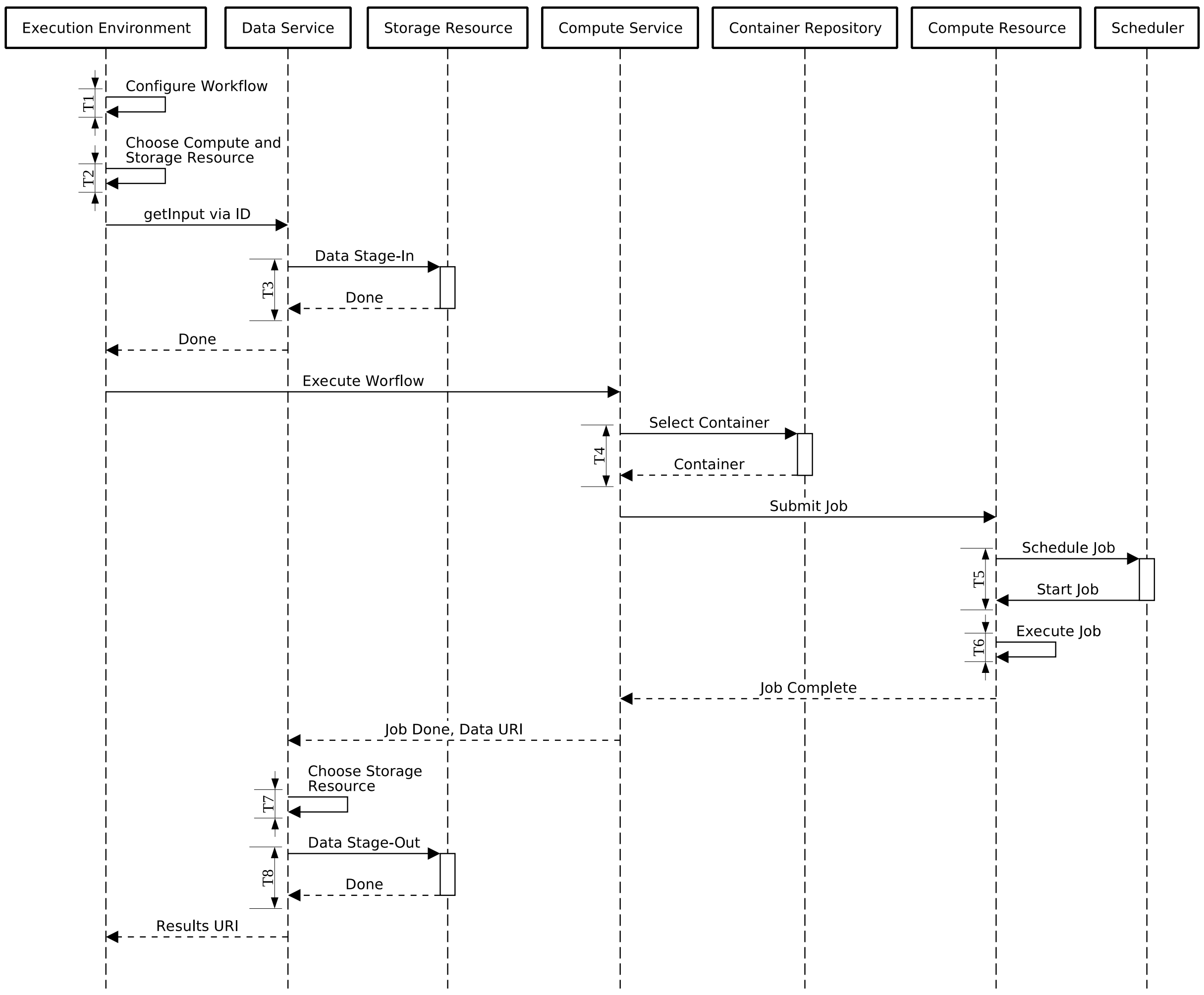

***Figure 1**: Sequence diagram describing the steps involved in the execution of a typical application scenario*

Table 2 summarises the various identified times which we will use as performance measurands.

***Table 2**: Description of the PROCESS measurands*

TX	Name	Description
T1	Configuration	Time to configure the workflow for the application
T2	Deployment Strategy	Time to select appropriate storage and computing site
T3	Stage-In	Time to transfer data from source to selected storage site
T4	Container selection	Time to select specified container for the workflow from repository
T5	Schedule	Time the submitted job spends in queue
T6	Execution time	Time spent executing the job on the compute resource
T7	Stage-Out Strategy	Time to select appropriate storage site for output
T8	Stage-Out	Time to transfer result to storage site

Using the performance measurands listed in Table 2, we propose a three-steps approach to the modelling and performance prediction of the PROCESS infrastructure. **First**, we will show that the overhead of the PROCESS platform for a deployment on one site (initializing the micro-infrastructure and scheduling) is negligible. **Second**, since the deployment strategy of PROCESS is to deploy every application in a container, we will show the weak scaling capabilities of PROCESS by deploying multiple containers with a split of the input data on one site. **Third**, since the goal is to achieve an exascale system solution, we will enable applications to scale by splitting the data and deploying containers across multiple sites of PROCESS.

We therefore describe three measurement scenarios:

Scenario 1: Single container – single site (Figure 2-a)

In this scenario we measure the execution time of processing the input sequentially within one running container. This container uses the maximum available quantity of compute resources PROCESS can use at one single site (e.g. use case 2 running only on one cluster).

Scenario 2: Multiple containers – single site (Figure 2-b)

In the second scenario we submit several containers on one cluster. Here, we either expect a speedup, since the container in scenario 1 may not have effectively utilized all available resources, or the same runtime as before, since the overhead to deploy more than one container in parallel should be minimal.

Scenario 3: Multiple containers – multiples sites (Figure 2-c)

The final scenario will involve several containers running in parallel on several different sites, with a corresponding split in the input data set. We expect significant speedup since multiple containers will be deployed on multiple sites.

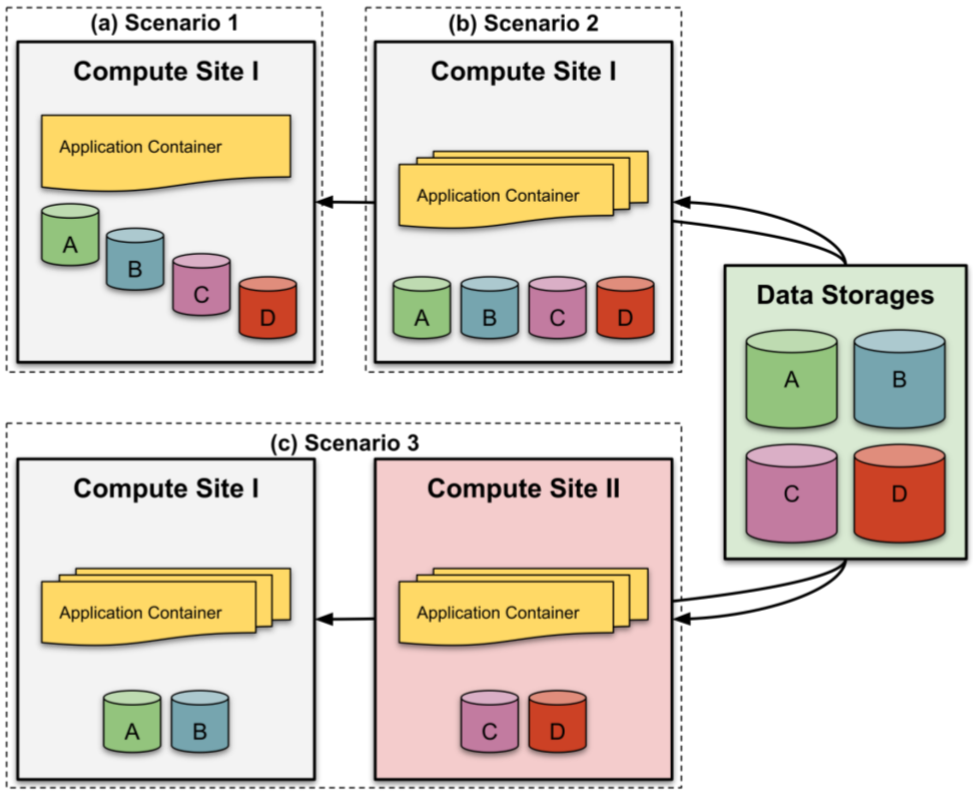

Figure 2*: Three measurement scenarios: (a) Single container – single site, (b) Multiple containers – single site, (c) Multiple containers – multiple sites. In all three scenarios Stage-In and Stage-Out will limit the overall system performances unless the wide area network data transfer issue is addressed.*

After evaluating these scenarios and measurements, we will present a generic performance model that will allow us to predict the scalability of the PROCESS infrastructure for a given application.

3 Development of a balanced Prediction Model

In this section we present our approach to determine the components of a simple predictive model for workflow performance on the PROCESS infrastructure.

3.1 Runtime Composition

Based on Figure 2 the total runtime of an application can be defined as follows:

$$\boldsymbol{Runtime = Overhead + Data\ Transfer + Scheduling + \ Execution\ Time}$$

Where:

$$\begin{aligned}&Overhead = T1 + T2 + T4 + T7\\&Data\ Transfer = T3 + T8\\&Scheduling = T5\\&Execution\ Time = T6\end{aligned}$$

The *overhead* component contains all overhead directly related to the PROCESS services. This includes selecting the appropriate resources for data access and compute in the Execution Environment, configuring the micro-architecture of LOBCDER for data access, fetching the application containers, and submitting the application to the selected resource using Rimrock.

To support exascale it is important that this *overhead* remains low for each submitted workflow and does not depend on the scale of the compute resource targeted by the services. We expect it to be orders of magnitude lower than the overhead associated with the remaining components, and therefore negligible.

The *data transfer, scheduling and execution time* components are mostly determined by factors outside of the control of PROCESS services, such as network capacity, queue waiting times, and how well a workflow performs and scales on a given resource. Nevertheless, having an estimate of the *data transfer and scheduling delay* is useful for selecting a resource to which a workflow should be submitted. If *execution time* estimates are available, this selection may be improved further, and a total *runtime* estimate may be provided to the user.

The *data transfer* component is mainly determined by two parts: the time required by DISPEL to perform pre-processing of the data (if any), and the time required to transfer the resulting data volume given the end-to-end transfer capacity between the storage and compute site. These two components may largely overlap if the data pre-processing is simple and can be performed on the fly, but for complex operations this may not be the case.

For the latter part, predicting large long-distance data transfers, a significant research has been performed over the last two decades. For example, [Liu2017] describes a model that predicts end-to-end data transfer times with high accuracy based on logs of the Globus transfer service.

Similarly, much research has been done with regard to estimating queue waiting times of HPC applications which dominates the *scheduling* component. For example, [Nurmi2007] describes a model that provides estimates with a high degree of accuracy and correctness for a large number of supercomputing sites.

For PROCESS we re-use this existing work to provide estimates for both the data transfer and scheduling components of the model.

Predicting the *execution time* is highly application specific and must be done separately for each of the use cases. It may be dependent on input datasets, application parameters and amount of resources used (number and type of cores, amount and speed memory, availability and type of GPUs, etc).

Strong scalability of the use case applications is expected to be limited well below exascale, as currently only very few applications are able to exploit the petascale level. To determine the limits of the strong scalability of the use case workflows, traditional performance benchmarking of the applications will be used for representative input data sets and parameters.

To circumvent strong scalability limits, we can exploit weak scalability, where multiple workflows are running at the same time to process different datasets. However, doing so may shift the bottleneck from the application to other sources, such as the data service, or local storage on the resources. Such limits can be discovered by performing weak scalability testing, both on a single site and across multiple sites.

Unfortunately, it requires substantial effort to create a complete and accurate model of the application behaviour for each of these use cases. Although users may be willing to perform some testing in advance, to fine-tune their applications, they are mostly interested in obtaining results. Therefore, highly accurate modelling of the application workflows is not required; instead, a rough estimate of the processing time is generally enough.

We will initially assume the user will provide an estimate for the execution time, as is customary on HPC systems. At a later stage, this estimate may be refined based on easily observed parameters, such as input data size and quantity of resources used, which may be extracted from the logs of previous runs of the given workflow. A significant amount of research has gone into estimating application execution time based on limited information. For example, [Smith1998] present a technique that predicts application run times based on historical information for “similar” applications. Search techniques are used to automatically determine the best definition of similarity. In [Gaussier2015], a similar technique is used to fine-tune the execution time estimate provided by the user.

3.2 Model Verification

3.2.1 Benchmark Application

An artificial benchmark workflow will be created which allows configuration of the different aspects of a workflow, such as the sizes and locations of input and output data, pre- or post-processing requirements, the number and type of compute resources required, the execution time of the application, etc. This benchmark workflow can be used to test the functionality or the PROCESS services, determine the initial model values, and validate model predictions.

By choosing minimal values for *data transfer* and *execution time* (for example 0 bytes and 0 seconds) a lower bound for the runtime can be obtained and the overhead of the PROCESS services can be measured. By submitting large numbers of such workflows, the scalability of the services themselves can be tested.

Choosing large values for data transfer yields an initial estimate of the data transfer capacity between sites. Similarly, different pre-processing patterns can be tested, ranging from straightforward filtering or conversion to more complex operations such as mixing or transpositions, to create an initial estimate of the DISPEL overhead.

By varying the target resources of the workflow, an initial estimate of the scheduling delays at different locations can be made.

Once an initial model is available, this benchmark application can be used to validate it by comparing the error rates of the predictions against actual measurements. This will allow us to iteratively refine the model during the course of the project.

3.2.2 Use Case Workflows

As explained above, strong and weak scalability tests may be performed on the use case workflows to determine the limits to its scalability and the initial parameters of the execution time models. Once these parameters are available, an initial execution time model can be created, and its predictions can be verified using the logs of subsequent workflow runs. Consistently measuring the workflow performance and selected key parameters (such as input data size and type and number of resources used) allow the model to be refined further. By default, a simple placeholder model will be used by the PROCESS services. If necessary, a more detailed use case specific model may be created for each use case and provided upon workflow submission.

3.3 Conclusion

In this section we have described the components of a simple predictive model for workflows performance on the PROCESS infrastructure. The main goal of this model will be to verify that the overhead incurred by the PROCESS services (the sum of T1, T2, T4 and T7 in Figure 1) is negligible compared to the cost of data staging (T3 and T8), scheduling (T5) and execution (T6). Using this model, we will attempt to verify whether the proposed services are capable of scaling into the exascale range.

4 Measurements

In D3.2 we will report upon the initial measurement results of a test application and early versions of the PROCESS use cases. Thereupon, we will also specify which methods were applied and which points of interest defined in the previous chapter could be verified.

This section will be filled within
D3.2 and D3.3.

5 Application of the Prediction Model to actual Measurement Results and Conclusion

This section will be filled within
D3.2 and D3.3.

6 References

[PMO] Performance Modelling, David Henty, EPCC, The University of Edinburgh [online: http://www.archer.ac.uk/training/course-material/2018/07/ScaleMPI-MK/Slides/Performance Modelling.pdf]

[Liu2017] Liu, Z., Balaprakash, P., Kettimuthu, R. and Foster, I., 2017, June. Explaining wide area data transfer performance. In *Proceedings of the 26th International Symposium on High-Performance Parallel and Distributed Computing* (pp. 167-178). ACM.

[Nurmi2007] Nurmi, D., Brevik, J. and Wolski, R., 2007, June. QBETS: queue bounds estimation from time series. In *Workshop on Job Scheduling Strategies for Parallel Processing* (pp. 76-101). Springer, Berlin, Heidelberg.

[Smith1998] Smith, W., Foster, I. and Taylor, V., 1998, March. Predicting application run times using historical information. In *Workshop on Job Scheduling Strategies for Parallel Processing* (pp. 122-142). Springer, Berlin, Heidelberg.

[Gaussier2015] Gaussier, E., Glesser, D., Reis, V. and Trystram, D., 2015, November. Improving backfilling by using machine learning to predict running times. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (p. 64). ACM.