Geosci. Model Dev., 12, 3001–3015, 2019
https://doi.org/10.5194/gmd-12-3001-2019

Geoscientific
Model Development
Open Access

# Scientific workflows applied to the coupling of a continuum (Elmer v8.3) and a discrete element (HiDEM v1.0) ice dynamic model

**Shahbaz Memon**[1,2], **Dorothée Vallot**[3], **Thomas Zwinger**[4], **Jan Åström**[4], **Helmut Neukirchen**[2], **Morris Riedel**[1,2], and **Matthias Book**[2]

[1]Jülich Supercomputing Centre, Forschungszentrum Jülich, Leo-Brandt Straße, 52428 Jülich, Germany
[2]Faculty of Industrial Engineering, Mechanical Engineering and Computer Science, University of Iceland, Reykjavik, Iceland
[3]Department of Earth Sciences, Uppsala University, Uppsala, Sweden
[4]CSC – IT Center for Science Ltd., Espoo, Finland

**Correspondence:** Shahbaz Memon (m.memon@fz-juelich.de)

Received: 25 June 2018 – Discussion started: 20 July 2018
Revised: 22 May 2019 – Accepted: 24 May 2019 – Published: 15 July 2019

**Abstract.** Scientific computing applications involving complex simulations and data-intensive processing are often composed of multiple tasks forming a workflow of computing jobs. Scientific communities running such applications on computing resources often find it cumbersome to manage and monitor the execution of these tasks and their associated data. These workflow implementations usually add overhead by introducing unnecessary input/output (I/O) for coupling the models and can lead to sub-optimal CPU utilization. Furthermore, running these workflow implementations in different environments requires significant adaptation efforts, which can hinder the reproducibility of the underlying science. High-level scientific workflow management systems (WMS) can be used to automate and simplify complex task structures by providing tooling for the composition and execution of workflows – even across distributed and heterogeneous computing environments. The WMS approach allows users to focus on the underlying high-level workflow and avoid low-level pitfalls that would lead to non-optimal resource usage while still allowing the workflow to remain portable between different computing environments. As a case study, we apply the UNICORE workflow management system to enable the coupling of a glacier flow model and calving model which contain many tasks and dependencies, ranging from pre-processing and data management to repetitive executions in heterogeneous high-performance computing (HPC) resource environments. Using the UNICORE workflow management system, the composition, management, and execution of the glacier modelling workflow becomes easier with respect to usage, monitoring, maintenance, reusability, portability, and reproducibility in different environments and by different user groups. Last but not least, the workflow helps to speed the runs up by reducing model coupling I/O overhead and it optimizes CPU utilization by avoiding idle CPU cores and running the models in a distributed way on the HPC cluster that best fits the characteristics of each model.

## 1 Introduction

The complexity of glaciological systems is increasingly reflected by the physical models used to describe the processes acting on different temporal and spatial scales. Addressing these complexities inevitably involves the combination of different sub-models into a single simulation that encompasses multiple tasks executed in a distributed computing facility. A particularly good example of such a combination is the simulation of calving behaviour at the front of glaciers that combines continuum model and discrete element model simulations. These computational tasks are connected to each other to form a "scientific workflow": the composition and execution of multiple data processing steps as defined by the requirements of the scientific application concerned.

Carrying out the discrete calving and ice flow modelling dynamically and as one workflow instance becomes very laborious without an automated mechanism. The analysis of such a workflow in more detail reveals that there are even more steps than just the two model elements mentioned above. These include, for instance, pre- and post-processing, job dependency management, job submission, monitoring, conditional invocations, and multi-site data management. These steps can be cumbersome for a user; for example, if any step produces an unexpected output, which can cause the whole workflow to fail, this may require the user to re-run the whole workflow. These issues often indicate that workflow implementation is sub-optimal as it requires coupling overhead (such as unnecessary I/O), or because a high number of CPU cores is allocated; this suits the most CPU-intensive task to be executed, but leaves cores idle when other tasks are executed that are less CPU-intensive or do not scale well. A workflow management system (WMS) allows the user to automate and ease the workflow management steps by means of abstraction, which not only increases usability, but also enhances portability to different computing platforms and, in turn, reproducibility of the scientific model runs. In our case, it also allowed us to focus on performance aspects, thus enabling a reduction in coupling I/O overhead and an optimization of the CPU utilization.

The main contribution of this article is to identify the workflow problems that need to be solved with respect to coupling a glacier continuum model and a discrete element model in an optimized way, in order to elicit corresponding requirements that address – among others – portability, performance improvements, and CPU utilization, and to implement an automated workflow based on the UNICORE (Streit et al., 2010) distributed computing middleware, in particular using the UNICORE workflow management system (Memon et al., 2007). We demonstrate this by combining ice flow modelling and discrete calving into a high-level easy-to-use and performance-optimized scientific workflow.

This article is structured as follows: Sect. 2 provides a discussion on the state of the art in glacier calving modelling and on scientific workflows. The targeted glacier modelling case study is presented in Sect. 3.1, which describes a workflow baseline used for model coupling and the applications used for execution. Next, the creation of a good solution is demonstrated, by first identifying the requirements to solve these problems (Sect. 4.1), followed by the creation of an improved matching workflow design (Sect. 4.2), and finally the implementation of the workflow (Sect. 4.3). The implemented workflow is evaluated in Sect. 5 from different perspectives, including a discussion on how the requirements from Sect. 4.1 have been fulfilled. Finally, a summary and an outlook conclude the article.

## 2 State of the art

### 2.1 Modelling and simulating the calving of a glacier

The calving behaviour at the front of glaciers is still a largely unresolved topic in modern theoretical glaciology. The core of the problem is that ice as a material shows different behaviour, depending on the timescale on which the forces are applied (Greve and Blatter, 2009). The everyday experience is that ice is a brittle solid body (e.g. breaking icicles). Such behaviour is observed if the reaction to a force is in the range of seconds to minutes. However, theoretical glaciology in recent years has rather dealt with the long-term (i.e. beyond minutes to millennia) behaviour of glaciers and ice sheets, where ice shows the property of a strong non-linear, shear thinning fluid (Greve and Blatter, 2009). This leads to a description of long-term ice flow dynamics in classical ice sheet and glacier dynamics models (e.g. Gagliardini et al., 2013) in terms of thermo-mechanically coupled non-Newtonian Stokes flow continuum models.

In stark contrast to such a description, the process of cracking or calving (i.e. the complete failure of ice fronts) is an inherently discontinuous process, that – if addressed in a physically correct way – requires a completely different model approach. Models adopting a discontinuous approach have been developed in recent years (e.g. Åström et al., 2013; Åström et al., 2014; Bassis and Jacobs, 2013). These models describe the glacier as discrete particles connected by elastic beams that can be dissolved if a certain critical strain is exceeded, and are therefore able to mimic the elastic as well as the brittle behaviour of ice. The size of these model particles (in the range of metres), which need to resolve a whole glacier of several cubic kilometres, inherently demands large computational resources. In addition, the characteristic speeds of advancing cracks is close to the speed of sound, which, in combination with the small spatial resolution, imposes a maximum allowed time-step size of a fraction of a second.

In other words, the combination of a discrete element and a continuum ice flow model is a temporal multi-scale problem, where the former basically describes an instant change of geometry or rheology for the latter. This means that both models need to be run in a sequential manner, with several repetitions. This defines a workflow of two model components that strongly differ in computational demand. In addition, these two model components have to effectively and efficiently exchange data – namely, the new geometries either changed by flow deformation or by calving as well as damage caused by fracturing.

### 2.2 Scientific workflows

A scientific workflow can be defined as the composition and execution of multiple data processing steps as required by a scientific computing application. Such a workflow captures a series of analytical steps of computational experiments to "aid the scientific discovery process through the combination of scientific data management, analysis, simulation, and visualization" (Barker and van Hemert, 2008). Conceptually, scientific workflows can be considered (and are typically visualized) as graphs consisting of nodes representing individual tasks and constructs, and edges representing different types of associations between nodes, such as sequential or conditional execution.

Carrying out the discrete calving and ice flow model simulations becomes complex as multiple parallel high-performance computing (HPC) applications are involved, especially if there are tasks in the workflow that consist of pre- and post-processing phases, and require multi-site and iterative job and data management functions. The overall scenario may easily become unmanageable, and the workflow management might be prone to errors, failures, poor reproducibility, and sub-optimal HPC resource usage.

A workflow scenario such as this will be even more challenging when some parts are launched on heterogeneous resource management systems equipped with different file systems, different data transfer mechanisms, and different job submission systems. In our case, for example, two different HPC clusters with different characteristics are simultaneously used: one for the ice flow modelling and another one for the discrete element (i.e. calving) modelling executions.

These workflow challenges can be addressed by a workflow management system (WMS) that is capable of managing the complex dependencies of many job steps with multiple conditional and nested constructs. Several scientific WMSs have been developed to automate complex applications from multi-disciplinary backgrounds. Ferreira da Silva et al. (2017) comprehensively categorized different workflow management systems according to the type of execution scenario and capabilities they offer, and also identified their specialized scientific domain. One example is Taverna (Wolstencroft et al., 2013), which in principle is a general WMS, but is significantly driven by bio-informatics communities with the need for high-throughput computing (HTC)-driven "-omics" analyses (proteomics, transcriptomics, etc.) and thus lacks the distinct support of cutting-edge HPC systems such as those used in our case study. In a similar manner, the Southern California Earthquake Center (SCEC) Earthworks Portal, a part of the US infrastructure Extreme Science and Engineering Discovery Environment (XSEDE), adopts two other HTC-related workflow management systems, namely Pegasus (Deelman et al., 2015) and DAGMan (Frey, 2003). Pegasus itself is just a component on top of DAGMan that, in turn, is based on the HTCondor middleware for HTC, which in our review did not really meet the full capabilities required for HPC in general or the particular capabilities needed for the large supercomputers used in our study.

Our scenario from the domain of glaciology using HPC technology includes a complex workflow graph; therefore, it is not easily manageable for users with limited expertise concerning HPC environments. Hence, it is important to implement the glacier modelling case study with a WMS that provides a rich graphical front-end and simultaneously offers a generic and seamless execution and monitoring of applications on HPC-based infrastructures in particular. Considering this requirement, the glacier model coupling case study is automated via our standards-based workflow management system (Memon et al., 2007), which is a part of the Uniform Interface to Computing Resources (UNICORE) (Streit et al., 2010) distributed computing middleware. It is specifically designed to support HPC applications deployed in a massively parallel environment. As described later in Sect. 4.3, our WMS for UNICORE provides a rich graphical interface for the composition, management, and monitoring of scientific workflows by users with different levels of system expertise.

## 3 Case study: Kronebreen glacier simulation

Coupling a continuum ice flow model and a particle-based calving model of the Kronebreen glacier provides a well-suited case study for the application of a WMS. More details on the scientific background of such a coupling and on the models are presented in Vallot et al. (2018). In Vallot et al. (2018), a series of key processes (ice flow, surface and subglacial hydrology, ocean water mixing, undercutting at the front, and finally ice calving in the ocean) were simulated by a sequence of different models in a one-way coupling. The aim was to show the feasibility of the coupling by comparing observations and historical data to simulation results and to identify the resulting interactions via this global approach. In this article, we introduce a full coupling that can be used for prognostic simulations. To simplify the problem we only use two models: an ice flow model (Elmer/Ice) and a discrete particle model (the Helsinki Discrete Element Model – HiDEM). In the following, we describe the software executables and the underlying workflow.

### 3.1 Conceptual scheme

Kronebreen is a tidewater glacier (ice flows directly into the ocean) and is one of the fastest-flowing glaciers of the Svalbard archipelago. After a period of stability, Kronebreen glacier started to retreat in 2011 and has continued since then. This glacier has been extensively studied (e.g. Kääb et al., 2005; Luckman et al., 2015; Nuth et al., 2012; van Pelt and Kohler, 2015; Schellenberger et al., 2015; Vallot et al., 2017), partly due to its location (close to a research station) and its interesting behaviour in terms of sliding and calving. For this reason, it is a good candidate for the present study. The aim is to reproduce both continuous (ice flow) and discrete (calving) processes using a finite element model (FEM) and a first-principle ice fracture model respectively.

## 3.2 Applications: meshing tools, continuum, and discrete ice dynamics model

Within our application, we use the continuum model Elmer/Ice (Gagliardini et al., 2013), and the discrete calving model HiDEM (Åström et al., 2013). Both codes can be run on large parallel HPC platforms. Implied by the physics that have to be addressed and by the modelling tools, the workflow contains three main applications that are part of the case study implementation:

1. Meshing: Gmsh (Geuzaine and Remacle, 2009) is the applied meshing tool used to provide the updated mesh for the continuum ice flow model run. It creates an updated footprint mesh in the horizontal plane that is further extruded and reshaped using the bedrock as well as free surface elevation to form the three-dimensional computing mesh for the ice dynamic simulation. Gmsh is an open source, versatile and scriptable meshing tool that perfectly matches the demands of being deployed within a workflow like this one. Gmsh applies a bottom-up geometry and meshing strategy, starting from outline points of the glacier, building a closed loop of its outline, and further creating a planar surface that is meshed in two dimensions.

2. Continuum modelling: Elmer/Ice (Gagliardini et al., 2013) is the open source ice sheet model used for computing the long-term dynamics of the glacier. Elmer/Ice is based on the multi-physics package Elmer (Råback et al., 2018), an open source finite element code developed by CSC – IT Center for Science Ltd. Elmer/Ice is able to utilize parallel processing, applying the message passing interface (MPI) paradigm (Message Passing Interface Forum, 2012) that uses messages to exchange data between nodes of a distributed parallel processing environment, and – for certain solver implementations – OpenMP (Dagum and Menon, 1998) for shared-memory parallel processing using multi-threading. Elmer also provides the `ElmerGrid` executable that can be used to convert the mesh created by Gmsh and at the same time perform a domain decomposition on the footprint, using the METIS library. The solver executable `ElmerSolver` has a built-in feature to internally extrude the given footprint mesh into layered columns of prisms and impose the given surface and bedrock elevation to form the volume of the glacier. Elmer is built on a shared library concept, meaning all solver modules are loaded during runtime. This enables the easy deployment of user-written functions and solvers through an application program interface (API).

3. Discrete modelling: HiDEM (Helsinki Discrete Element Model) (Åström et al., 2013) is a discrete element model that represents the glacier as mass-points connected by massless beams that are allowed to break if a given threshold is exceeded. An additionally applied repelling potential based on distance guarantees a non-intersection of compressed loose particles. Solving Newton's equation on such a set-up, it is possible to realistically reproduce the behaviour of fracturing. The downside of this approach is the high computational power demand, as the several cubic kilometre large glacier is discretized in pieces of a few tens of cubic metres. Furthermore, the time-step size for the simulation is imposed by the ratio of the speed of sound to the typical length of the discretized particles, which clearly falls below seconds. Hence, despite the fact that the code is utilizing massive parallel computing using the MPI paradigm, only a few minutes to hours of physical time can be computed, even on a huge HPC cluster. HiDEM receives the initial geometry from Elmer/Ice, in the form of gridded data over a limited area at the tongue of the glacier, and also receives the basal friction coefficient distribution computed within the continuum ice flow model.

## 3.3 Initial base workflow

The continuum ice flow model and the discrete calving model need to be coupled, which leads to a scientific workflow. A baseline version of the workflow was initially realized by a Bash shell script that calls the above executables as well as additional Python helper code, and performs all of the required management operations using shell script commands. The underlying workflow is as follows:

### 3.3.1 Step 1: generate the mesh for Elmer/Ice

At $t = t_i$, the front position, $F_i$, and the glacier contour (two-dimensional boundaries of the glacier), Cont, are given as input to create the mesh, $M_i$. This determines the domain of the glacier for the ice flow model Elmer/Ice. A Python script invokes the meshing executable `gmsh` to build the mesh from the contour and the front position, and invokes `ElmerGrid` to convert it into Elmer format. In this particular application, the mesh is then split into 16 partitions. This step runs as a single-threaded application; with respect to resource requirements, mesh generation is not very CPU-intensive (serial, i.e. 1 CPU core), but it consumes some storage space.

### 3.3.2 Step 2: ice flow modelling and conversion to the HiDEM domain

The continuum ice flow model is executed using the `ElmerSolver` application which is an MPI-based implementation and part of the Elmer application suite. The number of time steps ($N_t$) depends on the glacier and process studied as well as the spatial resolution. Here, we simulate $N_t = 11$ d in one run, which – using a time-step size of 1 d – corresponds to the update frequency of remote-sensing satellite data which were used for model validation.

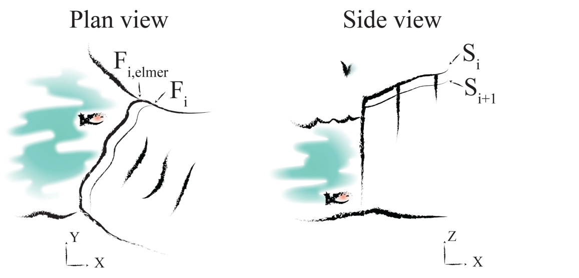

**Figure 1.** Conceptual plan and side view of an Elmer/Ice transient run. The initial front $F_i$ and surface $S_i$ are evolved to a new front position $F_{i,\mathrm{elmer}}$ and a new surface elevation $S_{i+1}$.

As the basal boundary condition (BC), we assume a no-penetration condition with no basal melting nor accumulation and basal friction (currently Weertman friction law). The upper BC is defined as a stress-free surface and is able to evolve during the simulation following an advection equation forced by a surface mass balance. As the BC in contact with the ocean, the normal component of the stress vector is equal to the hydrostatic water pressure exerted by the ocean where ice is below sea level. The front is also able to evolve, using a Lagrangian scheme (i.e. mesh velocity equal to ice flow velocity corrected by melting). The temperature profile in the ice and lateral BC are prescribed but can be changed easily. The ice flows as a non-linear isotropic viscous fluid following Glen's flow law (e.g. Cuffey and Paterson, 2010), and the Stokes equations for an incompressible fluid are solved over the ice volume. Elmer/Ice executes a solver input file, $\mathtt{SIF}_i$, with the above-mentioned parameters.

After the simulation, the glacier front has a new position $F_{i,\mathrm{elmer}}$, a new surface elevation $S_{i+1}$ (see Fig. 1), and a map of basal friction coefficients (determined by a linear sliding law). These form the output of Elmer/Ice and the input to HiDEM. Elmer/Ice can be sped up by parallel processing, but is not as CPU-intensive as HiDEM; hence, only 16 CPU cores are used for Kronebreen glacier in this part of the workflow.

The output format of Elmer/Ice does not match the input format of HiDEM, and HiDEM does not need to process the whole glacier, only the front that is relevant for calving. Hence, a conversion step runs a set of helper scripts ("Elmer to HiDEM") implemented in Python. This step performs a conversion of the output of `ElmerSolver` to the HiDEM grid (10 m × 10 m in our case) that is used for the calving front of the glacier. Elevations are offset so that the minimum bed elevation is equal to zero. It also includes places with no ice where the surface elevation is equal to the bed elevation. This conversion step creates a text input file for HiDEM, $\mathrm{Pin}_i$, with coordinates, surface, bed and basal friction coefficients. This conversion is performed on a single CPU.

### 3.3.3 Step 3: discrete particle modelling and conversion to the Elmer/Ice domain

The converted output from Elmer/Ice is passed to HiDEM. The implementation of HiDEM requires an MPI environment (560 cores running the MPI processes in our case), because in comparison to the other steps, it consumes the biggest share of CPU hours due to the high spatial and temporal resolution. The stress field in both models is a consequence of gravity acting on the specific ice geometry and therefore initially identical (differences arise only owing to the discretization method). Deviations between the models are only in the response to stress, i.e. the rheology. Elmer/Ice deformations are viscous, HiDEM elastic–brittle. This means that in the latter only elastic deformations (but no viscous deformations) are modelled and the transfer of viscous rheology parameters from Elmer/Ice to HiDEM can be omitted. The fracture timescale is determined by the ratio of the glacier's spatial dimension and the velocity of sound, which is usually in the sub-second range. Ice flow, in comparison, occurs on the scale of relaxation time, which is dominated by a large viscosity in relation to a significant smaller Young's modulus and is therefore in the range of hours and beyond – several orders of magnitude larger. Consequently, HiDEM results can be interpreted as instantaneous for the ice flow model, Elmer/Ice. Under this assumption, we scale down the friction parameters HiDEM receives from Elmer/Ice (in our case using the factor $10^{-4}$) so as to increase the sliding speeds and reduce the physical time (in our case 100 s) needed to evaluate the resulting fractures in order to avoid excess computations. Despite the scaling, the relative distribution of friction parameters is maintained. As neither the glacier geometry nor the value of ice density or gravity are altered, this simply accelerates, but does not significantly alter the general glacier dynamics; this leads to the same fraction pattern that would have been obtained with unscaled friction parameters. A new front position, $F_{i+1}$, is determined after the simulation. In this step, the data produced are rather large because of the high spatial resolution.

Once a HiDEM run is completed, the next step is to reconvert this data set to the Elmer/Ice format, so that the next iteration of the coupled glaciology models' workflow can begin. Again, a Python script ("HiDEM to Elmer") is used to convert the HiDEM output into a format that can be read by Gmsh and ElmerGrid. For this purpose, the new front position $F_{i+1}$ (after calving) and surface elevation $S_{i+1}$ are reintroduced into Step 1 at $t = t_{i+1}$. Again, the conversion is performed in serial execution. After this step, the workflow reiterates from Step 1.

## 4 Workflow

### 4.1 Requirements analysis

The problem analysis of the initial shell script-based workflow led to a set of requirements that aim at improving the workflow with respect to usability, adaptability, maintainability, portability, robustness, resource usage (I/O and CPU), and overall runtime. Based on the weaknesses of the initial workflow implementation, we particularly focused on reducing the overhead associated with the initial coupling approach by improving the overall runtime, optimizing the CPU resource usage, and coupling-related I/O, as well as the factors mentioned previously regarding enabling a uniform access to widen the scientific community's adoption of this glaciology workflow.

The requirements elicitation phase yielded the following requirements, which led to an improved design and implementation of the workflow (a summary of the requirements and a description of the UNICORE-based implementation are provided in Table 1):

– ***R1: readability and understandability***

  The continuous development, maintenance, and dissemination of a scientific application for collaboration requires that the implementation have a clean, clearly modularized, and system-independent code. The workflow implementation should not contain static resource-specific details or malformed data locations as they may lead to subsequent runtime task failures. As our case study consists of many independent applications related to each workflow task, it is important that the tasks are well-segregated and do not overlap. A well-segregated workflow not only helps the application developer to further enhance the application, but also to distribute the code in order to collaborate with a larger scientific community.

– ***R2: sequential pipeline***

  The execution of jobs in the workflow should be orchestrated in a sequential manner such that one job step should not commence unless all previous steps have been completed. This requirement envisages the whole scenario as a sequence of jobs that should connect all the scientific applications taking part in the glacier model coupling case study.

– ***R3: dynamic data injection***

  The data injection for any workflow job should be transparent and easy to express. This requirement refers to the provisioning of data sets to individual workflow steps: before a job is started, the required data needs to be available. Furthermore, dynamic data injection allows for the importation of data from various sources using different protocols. A data-transfer-agnostic access is an add-on to this requirement.

– ***R4: minimize coupling I/O***

  The cost of data sharing across the jobs of the workflow steps becomes high when the data are unnecessarily replicated across each of the job steps. This increases the use of storage space and negatively impacts the overall workflow footprint in terms of resource consumption, in particular with respect to I/O performance. Therefore, an adequate data sharing mechanism that minimizes the coupling-related I/O of all of the tasks should be available, which concurrently allows a simplified integration of data at application runtime. It will also facilitate optimal storage resource usage (e.g. of a parallel file system) in the target system. This is of particular importance when dealing with two different HPC clusters running different steps of the workflow, where data need to be exchanged between the HPC clusters.

– ***R5: minimize CPU resource consumption***

  The continuum ice flow model (Elmer/Ice) is less CPU resource-intensive than the calving model (HiDEM). This is due to very different spatial and temporal resolutions but also the models themselves, which require different amounts of computational resources (16 cores for the continuous ice flow model, and 560 cores for the discrete particle model). Getting access to CPU time on a small HPC cluster is typically easier than on big clusters; hence, the workflow will support the possibility of running these two significantly different steps on two different computing resources, which reduces the amount of CPU and queuing time needed on the larger cluster. If the executions are run on heterogeneous clusters, a layer of abstraction is needed that encapsulates the intricacies of different resource management systems (see R11). If the executions are run on the same cluster, the allocation of more cores than are actually used needs to be avoided (e.g. do not allocate 560 cores to run a 16-core Elmer/Ice job or even to execute a serial data conversion script).

– ***R6: parametric execution***

  In our case study, most of the job steps need to be executed in an iterative way. With every new iteration, input has to be extracted from a plain ASCII text file, called `n_list.txt`, which contains the surface velocity data of some days of ice flow simulation. Here the requirement is to use the input data from the file and parameterize them for guiding the workflow iterations. Furthermore, the envisioned workflow implementation ensures that the number of resulting iterations should abide by the number of observations defined in the input file.

– ***R7: workflow composition and visual editing***

  It is more robust for users to have a graphical interface that allows them to visually program and manage scientific workflows. In the glacier modelling scenario there

**Table 1.** Summary of requirements and how the new workflow implementation addresses them.

| | Description | UNICORE-based realization |
|---|---|---|
| R1 | Readability and usability | URC workflow management and interface |
| R2 | Sequential pipeline | Workflow management and enactment |
| R3 | Dynamic data injection | Automatic data import and export |
| R4 | Data sharing across job steps | UNICORE's data management services |
| R5 | Resource-agnostic access | UNICORE's job management services |
| R6 | Parametric execution | Composite constructs and loops |
| R7 | Workflow composition and visual editing | URC workflow editor and widgets |
| R8 | Workflow tracing and monitoring | UNICORE's workflow tracing and management services |
| R9 | Workflow reproducibility | URC's project export wizard |
| R10 | Secure access | PKI, X.509, and mutual authentication |
| R11 | Execution environment independence | Job incarnation through XNJS and target system interface (TSI) |
| R12 | Data and variable configuration | Middleware-supported variable resolution |

are six main steps, each with different shell scripts and resource configurations; therefore, a graphical user interface (GUI) can be very useful for visual editing, composition, and automation of all of the steps.

– ***R8: workflow tracing and monitoring***

It should be possible to trace and monitor the whole workflow, including its sub-elements such as individual jobs. The extensive process of calving simulation may have to be aborted at some stage due to data or parameter anomalies. Therefore, it must be possible to interrupt the workflow at any point. Apart from this, the real-time status of the jobs managed by the workflow should be provided.

– ***R9: workflow reproducibility***

The workflow needs to support the reproduction of results, both by the original researchers and by third parties. If the workflow is carefully designed in a way that enables it to adopt different computing and data environments, it can be exported for reuse by a larger community. This includes not only exposing the workflow to a wider community on the same computational resource, but also running it in a completely different hardware or software environment (reusability, adaptability, portability, and maintainability).

– ***R10: secure access***

The workflow management system should be capable of providing an interface to let users run scientific workflows in a secure manner. This implies that adequate authentication and authorization need to be in place. This requirement further mandates that the workflow system be compatible with back-end computing clusters and existing production computing infrastructures.

– ***R11: execution platform independence***

This requirement supports a scenario which allows scientists to submit computations without knowledge of the parallel execution environment installed at the target computing resource. In our case, there are at least two different MPI execution environments involved, and thus two different MPI implementations. Another aspect is to abstract from the batch system used for job submission to the HPC cluster(s). The intended middleware abstraction should not require a user to know the target environment that is providing the actual execution.

– ***R12: data and variable configuration***

Configuring required data elements such as workflow-centric input and output locations, and shared applications' environment variables or constants across many job steps can reduce much workflow management and development overhead. This may allow for the design, execution, and debugging phases of many tasks to be carried out in a more efficient manner. Therefore, in terms of overall usability and application maintenance, this requirement is considered important for realizing the complex structure of connected tasks.

Any solution that addresses this set of requirements will make the scientific workflow usable for a wider set of communities working in glaciology.

## 4.2 Workflow design

Using the initial shell script-based workflow as a starting point and taking the requirements R1–R12 into account, this section discusses the design of the glacier model coupling workflow implementation. Figure 2 shows the workflow composition.

The data conversion tasks, such as "Elmer to HiDEM" and "HiDEM to Elmer" existed in the initial workflow implementation as part of the respective ElmerSolver and HiDEM jobs. As the latter are both resource-intensive (i.e. they run on multiple cores), whereas the data conversion tasks are serial and require less resources, it is inappropriate to reserve (and thus

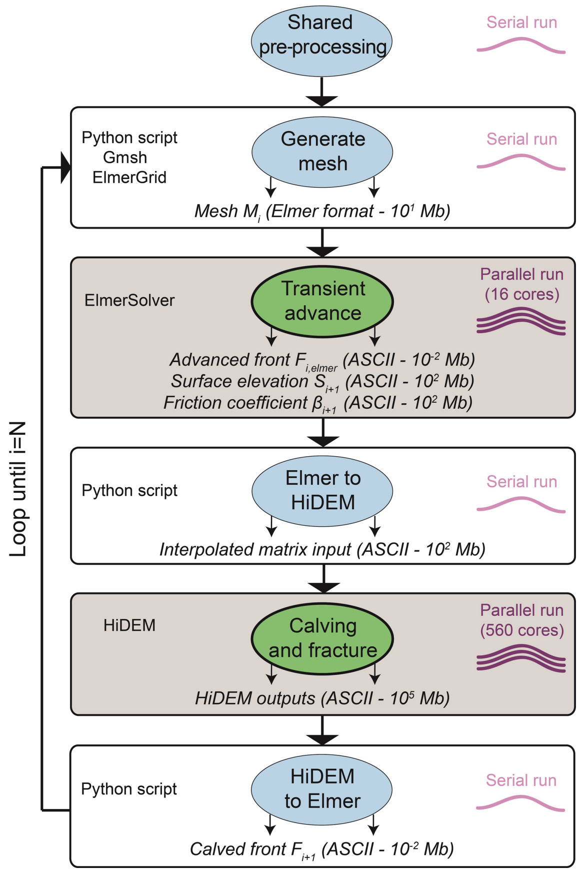
Shared pre-processing
Serial run
Python script
Gmsh
ElmerGrid
Generate mesh
Serial run
Mesh $M_i$ (Elmer format - $10^1$ Mb)
ElmerSolver
Transient advance
Parallel run (16 cores)
Advanced front $F_{i,elmer}$ (ASCII - $10^{-2}$ Mb)
Surface elevation $S_{i+1}$ (ASCII - $10^2$ Mb)
Friction coefficient $\beta_{i+1}$ (ASCII - $10^2$ Mb)
Python script
Elmer to HiDEM
Serial run
Interpolated matrix input (ASCII - $10^2$ Mb)
HiDEM
Calving and fracture
Parallel run (560 cores)
HiDEM outputs (ASCII - $10^5$ Mb)
Python script
HiDEM to Elmer
Serial run
Calved front $F_{i+1}$ (ASCII - $10^{-2}$ Mb)
Loop until i=N

**Figure 2.** Generic workflow layout with blocks representing steps.

waste) parallel resources for the serial data conversion. The separation of tasks in the workflow's design enables them to use only a single core, which is sufficient for their serial execution.

The step "shared preprocessing" is introduced as an additional task to manage the initialization phase of the workflow. It mainly provides the applications with the required initial input data sets and prepares shared output directories where the subsequent individual workflow steps accumulate intermediate and final results. In this step, the shared workflow variables are also initialized, and the required intermediate working directories are created.

## 4.3 Workflow implementation

This section describes the workflow implementation and its realization through UNICORE (Uniform Interface to Computing Resources, which includes a workflow engine). The UNICORE middleware is not only used for the development and automation, but for the processing and management of the entire workflow on deployed HPC resources. We have contributed to the creation of both UNICORE in general (Memon et al., 2007) and the workflow engine in particular (Memon et al., 2013b). This section briefly details the

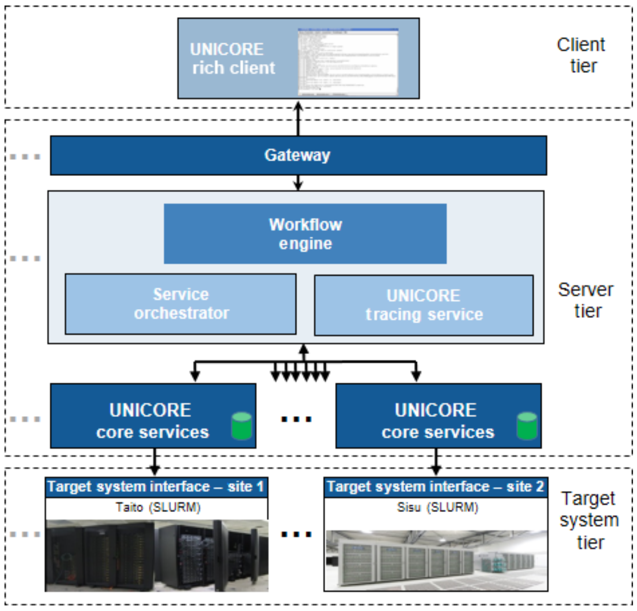

**Figure 3.** Multi-tiered UNICORE architecture.

UNICORE foundations, the workflow implementation, and concludes with the resource set-up and interaction.

### 4.3.1 UNICORE foundations

UNICORE (Streit et al., 2010) is a distributed computing middleware that provides abstractions for job submission and management on different kinds of job scheduling systems. Hence, jobs can be submitted to a cluster without needing to know about the internal job scheduling system used by that cluster. The abstraction is achieved through a unified set of interfaces that enable scientists to submit computation jobs without considering any intricacies of the underlying batch system. UNICORE takes care of the automatic translation of job requests to multiple target resource environments.

UNICORE provides a workflow system based on a service-oriented architecture (SOA), i.e. all of the main functional interfaces of the workflow system are exposed as web services. Figure 3 gives a holistic view of UNICORE's multi-layered architecture that is composed of client, server, and target system tiers. The client tier has two main variants, the UNICORE command-line client (UCC) and UNICORE rich client (URC). However, other third-party client applications such as scientific gateways, science portals and client APIs can also be integrated, if they comply with the provided server-side interfaces.

To address the goal of usability, we put emphasis on the URC, which is an Eclipse-based (Eclipse Foundation, 2013) client application implemented in Java. It provides users with a wide range of functionalities such as workflow management and monitoring, data download and upload to a remote cluster, a GUI for workflow editing, and resource and environment selection panels. For more details about the URC we refer to Demuth et al. (2010).

### 4.3.2 Workflow realization using UNICORE

Considering the complexity of the compute and data aspects, satisfying our requirements R1–R12 would take tremendous effort if no abstractions and high-level concepts (such as those provided by UNICORE) were used. Therefore, we employ the UNICORE workflow management system to automate the workflow of our case study in a high-level way.

To improve usability, the new, improved workflow was designed using the visual editor provided by the URC. The editor allows scientists to visually drag and drop different task types for different application types that may be enclosed in conditional structures. The supported task types are simplified, and small Bash shell scripts containing customized or generic applications can be executed remotely on user-specified resources.

Figure 4 shows the URC-based implementation of the workflow sketched in Fig. 2, outlining a sequence of the workflow tasks defined for our glacier modelling case study. The major steps described in Sect. 3.3 can be directly mapped to the tasks defined at the URC level. In addition to the initial workflow, we introduce the *prerun* step, wherein we declare constants for all of the workflow instances and also create a central output directory that is shared across all the jobs participating in the workflow. It also sets an initial input that contains the total number of iterations. Furthermore, while in the previous model runs, the conversion procedures were integrated from continuum to discrete model, they are now implemented as separate tasks within UNICORE's workflow implementation.

In the task definitions, a shared variable is required that contains the workflow output location that is to be used across all of the tasks. This is the only variable meant to be changed for a different user: if another user wants to run the same workflow on the same set of resources, e.g. the same HPC cluster, this single value has to be adjusted to the preferred file storage location.

Prior to the workflow execution by URC, the user has to carry out the following: (1) configure the target site that each task runs on; (2) specify the extent of computing resources it requires; and (3) provide a list of input and output files involved. Once the tasks are prepared, the workflow can be submitted for execution on (remote) HPC clusters. During the workflow execution phase, the sequence of running tasks follows the workflow graph specified by the user.

Listing 1 shows two code snippets from the URC environment. The first snippet shows the kernel of the particle-to-Elmer task, which simply invokes the Python executable. Batch system-specific components are added by the WMS (lines 2–5). The second snippet is the common code section (lines 7–12) that fetches the value of the last iteration required to process the data of the current iteration. This common code snippet must be present in all of the workflow steps (including the iteration blocks) except the *prerun* step. Practically, this is just a syntactic overhead, and is considered negligible when the number of affected steps is small.

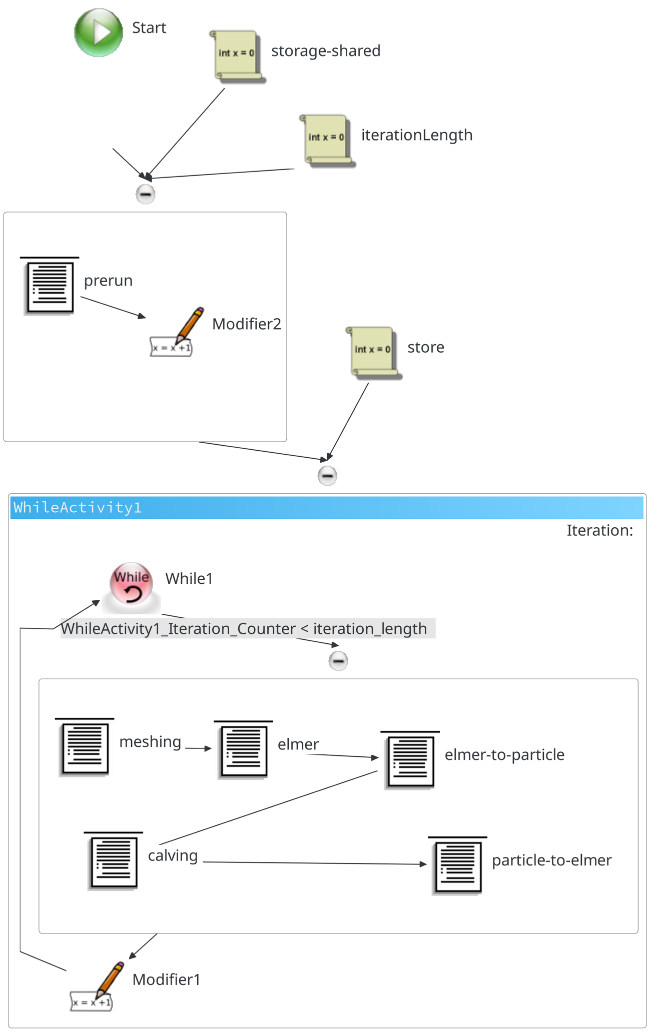

**Figure 4.** Workflow implementation in the URC workbench.

### 4.3.3 Resource set-up and interaction scenario

Our new and improved workflow requires the deployment of separate UNICORE server and client instances. The server-side deployment spans two production clusters at CSC in Finland, namely the *Taito* cluster for smaller jobs such as Elmer/Ice and the bigger *Sisu* cluster for massively parallel jobs such as HiDEM. At both sites, the UNICORE instances were deployed separately. These sites already have SLURM available as a resource management system, but with different hardware configurations: *Taito* has heterogeneous node groups with varying capabilities and CPU layouts (Intel's Haswell and Sandy Bridge processors), whereas *Sisu* has a symmetric configuration with all of the nodes providing same

```

particle-to-elmer:
    # creating task-specific folders
    python $JOBWD/input/Python_particle/ParticleToElmer.py $nb $JOBWD
    # accumulate output and move to the main workflow output

common code:
    if [ $currentctr -gt 1 ]; then
      nrbefore=`sed -n -e "$((COUNTER))p" $JOBWD/input/n_list.bak`
      nb_before=`awk '{print $1}' <<< $nrbefore`
      firstIteration=false
    fi
```

**Listing 1.** Excerpt of the particle-to-Elmer tasks and the common script snippets implemented in URC.

number of processing, data, and memory resources. Furthermore, as some sort of master, a shared UNICORE workflow management and a resource broker instance have been deployed on a cloud computing instance running at Jülich Supercomputing Centre (JSC) in Germany.

The resources at CSC needed to have the respective applications installed to support the complete workflow execution, i.e. on *Taito*, the Elmer suite with the Elmer/Ice glaciology extension was installed, whereas the particle calving application (HiDEM) was provided on *Sisu*. In addition to these executables, a Python environment had to be available on both systems for running the Elmer-to-particle (and particle-to-Elmer) conversion scripts.

The workflow development and remote execution is managed by the user through the UNICORE rich client (URC). Thus, it is required to have the URC installed on the user side. To obtain access to the remote UNICORE sites at CSC, the user has to acquire X.509 credentials and trusted certificates of the server instances. Utilizing these credentials, the user can interact with all of the UNICORE server instances of the infrastructure that they have access to.

After the credentials are set up, the user has to add sites that will be used for the workflow execution. For this, only a single location of the discovery service called "Registry" is provided. If the user's identity is known to the Registry instance and all the concerned compute sites, then these sites will be available for execution. Following the discovery service inclusion, the workflow development and management can be performed easily.

Figure 5 shows the output of the calving task that is the last step of each workflow iteration. Without the UNICORE-based implementation, it was difficult to manage the overall makespan of this step and the whole workflow in general. After transforming it using the UNICORE system, the management became easier and seamless as the calving tasks were only invoked when the preceding phases (such as coupling and data conversion) and iterations were completed successfully, i.e. when they could provide reasonable input to the

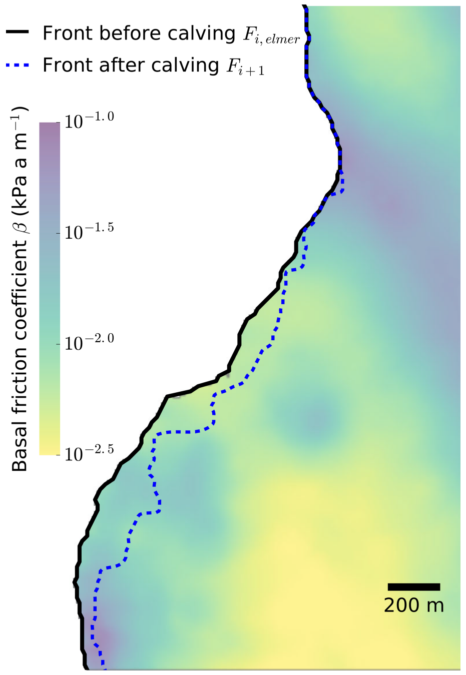
Front before calving $F_{i,elmer}$
Front after calving $F_{i+1}$
Basal friction coefficient $\beta$ (kPa a m$^{-1}$)
$10^{-1.0}$
$10^{-1.5}$
$10^{-2.0}$
$10^{-2.5}$
200 m

**Figure 5.** Basal friction coefficient, $\beta$, and new position of the front, $F_{i+1}$, after calving from the HiDEM simulation generated through the UNICORE-based implementation.

computationally expensive HiDEM executable. In addition to the reduced makespan effort, the UNICORE-based implementation enables users to remotely intervene in the running workflow instance by allowing them to access the individual task's execution state, working directory, and multi-site shared storages.

## 5 Discussion

This section discusses the most vital elements of the glacier model coupling case study to show that the main goal was achieved, i.e. a coupling of two different models dynamically and simplified access to distributed HPC resources.

In the implementation using UNICORE, we use the same application environment and data set as before the introduction of a WMS. Therefore, this discussion does not cover the application performance and scalability, but rather the overall workflow effectiveness, robustness, and usability.

### 5.1 Fulfilment of requirements

In this section, we show how the requirements presented in Sect. 4.1 have been fulfilled through the use of the UNICORE workflow management system. Table 1 lists each of the requirements and briefly explains its realization in the UNICORE-based version. The details are as follows:

Requirements *R1 – readability and usability* and *R7 – workflow composition and visual editing* are addressed through the URC client as it comes with a rich workflow editor that allows simplified composition and association of workflow tasks in a user-friendly manner.

The UNICORE WMS provides sequential access by enforcing a barrier on a workflow task that is about to be processed until its preceding task completes successfully. This supports requirement *R2 – sequential pipeline*.

The workflow's data management is considered to be an essential requirement for any data-intensive application. It is typically either a remote data transfer or data movement within the file system used by that computational resource. The workflow management system should not bother the user with this. In our case, the UNICORE atomic services take care of any data movement to a third-party data space or a local cluster; the user is only expected to specify the source and target file locations. After the workflow has been submitted, the required data transfers are carried out by the UNICORE middleware. This functionality supports requirements *R3 – dynamic data injection* and *R4 – minimize coupling I/O*.

For the glacier modelling workflow, the UNICORE-based implementation executes steps 2–6 of the workflow in a *while* loop until a certain number of observations has been reached. As the observations are stored in a file, they need to be processed and the values need to be loaded to UNICORE's *while* loop variable store. Each time the workflow instance is created and submitted, the loop construct loads the file called `n_list.txt` and takes each observation from that file to run the underlying steps in a parametric way. This feature supports requirement *R6 – parametric execution*.

The UNICORE-based glacier model coupling workflow uses the computing resources deployed on CSC's *Sisu* and *Taito* clusters. If a future user of this application intends to deploy and run the workflow in a different resource and application environment or with a different number of cores, this will be possible with minimal effort: the UNICORE atomic services provide a layer of abstraction over execution environments and batch systems, which fulfils requirements *R11 – execution platform independence* and *R5 – minimize CPU resource consumption*. The latter requirement is also fulfilled by splitting the previously monolithic job submission into separate job submissions, thus allowing for the specification of the exact number of cores needed for each job and preventing idle CPU cores that are reserved but in fact never used.

If another user is interested in using the UNICORE-based implementation, URC provides a feature to export the workflow in a reproducible format that can be reused by other users. This supports requirement *R9 – workflow reproducibility* (more details on workflow reproducibility are discussed later in Sect. 5.8).

The URC interface allows users to specify any application-, data- and environment-specific variables, scoped either to one task or a group of tasks. To enhance and simplify our new workflow implementation, a number of workflow-wide and application-specific variables were used. During workflow runtime, they are resolved without needing any user intervention. This addresses requirement *R12 – data and variable configuration*.

Requirement *R10 – secure access* is essential as the workflow will have access to large and precious compute resources, for which the UNICORE-based deployment ensures secure interaction between the user and all of the services they communicate with, such as workflow executions and access to storage and data. Users accessing any remote UNICORE-based services are required to possess X.509 credentials in order to use these services.

Finally, to compose, manage, and monitor workflow submissions interactively, URC provides a separate visual interface to edit or create individual workflow tasks or monitor running workflows and their jobs. This supports requirement *R8 – workflow tracing and monitoring*.

### 5.2 Middleware deployment overhead

While UNICORE is powerful and enabled the optimization of our workflow, having that additional layer may introduce some overhead: the provider of the computational resources has to ensure the availability of UNICORE server-side. Maintaining a server-side deployment requires a dedicated server that manages workflow jobs. Conversely, the URC is easy to use due to its GUI and does not have any significant installation overhead – it is Java-based and thus easily usable on any hardware platform.

### 5.3 Modularization

The UNICORE-based implementation using URC allows us to cleanly separate the tasks in a modular way, which enables us to individually monitor and manage tasks even while they are in the execution phase. The complete workflow management can be performed interactively and visually through the URC's GUI. Our experience is that using the URC is less error-prone than the purely shell-script-based approach.

### 5.4 Data transfer and management

UNICORE significantly eases data handling: for example, during the UNICORE-based workflow development and testing phase we ran one application instance in Germany (JSC) the other in Finland (CSC) with respective Elmer/Ice and HiDEM deployments, i.e. the workflow execution was distributed and the associated data had to be transferred back and forth between both sites. With the shell-script-based implementation, we found that the inter-task input/output (I/O) of the workflow was not easy to manage due to manually configured locations, so it was prone to multiple data transfer errors during the data staging phase. In contrast, the UNICORE-based approach takes care of the data movement automatically, and only the input and output files and their physical or logical addresses have to be declared in the beginning. Furthermore, through the UNICORE-based implementation, the user can easily connect outputs of one task as inputs to other, which implies that the connecting task will not begin unless the preceding task is successfully finished and the desired outputs are produced.

### 5.5 Efficient resource utilization

Using our improved workflow allowed us to optimize CPU utilization of running the models and data conversion and to speed up the I/O needed for model coupling.

The ratio of computational resources needed between HiDEM and Elmer/Ice is about 10 : 1. Hence, when running the same workflow as a single job submission (allocating the number of CPU cores needed for HiDEM), 90 % of the CPU cores in the Elmer/Ice stage would go idle, which would lead to extremely bad performance (not in terms of wall-clock time, but in terms of efficiency). In a similar manner, the data conversion steps in the workflow are less computationally intensive, and if attached to any of the Elmer or HiDEM job submission, could be very inefficient in terms of resource utilization. This is avoided by using UNICORE, which provisions each workflow task with a separate resource requirement specification and also enables different computing platforms (SLURM, PBS, etc.) to be combined into a single workflow.

Our workflow approach minimizes the I/O consumption by using UNICORE's internal workflow storage management services, which make data available (in the scope of a single infrastructure) to the next task without creating any explicit copy of the data set. Yet, UNICORE is flexible enough to also support distributed scenarios, by automatically transferring data across geographically distributed workflow tasks in a secure way – as described previously with respect to using resources at both JSC in Germany and CSC in Finland. The glacier model coupling case study intensively uses UNICORE's managed workflow storage, which organizes the individual workflow instances and their output data. In a resource-conservative environment, where the secondary storage, although not costly, is limited and regulated on the basis of site-constrained user quotas, the workflow-wide storage services proved to be adequate while supporting the complex data management requirements.

Usage of shared variables spanning all workflow tasks is essential to the glacier model coupling workflow implementation. In the UNICORE-based approach, the *prerun* job encapsulates the creation of shared variables for managing a workflow-wide structure useful for arranging the input and output data in a single location. This is realized through a single job that runs on the cluster's login node with a low processor and memory footprint.

### 5.6 Extendable workflow structure

In practice, enhancements or the addition of new scientific methods to an existing workflow are inevitable. In the UNICORE-based workflow implementation, in contrast, the editing and validation of the individual workflow tasks or the whole structure can easily be performed. This process occurs before the whole workflow is submitted for execution on HPC resources. Furthermore, if there are any enhancements to be performed after the workflow request is submitted, the running workflow instance can even be put on hold for intermittent script updates and restarted later.

The glacier model coupling model uses Elmer/Ice and HiDEM to analyse the complete scenario in the form of one structured recipe. Our approach adds flexibility due to the fact that the models are interchangeable. This means that if glaciological models other than Elmer/Ice or HiDEM are to be used, they can easily be integrated into the existing workflow structure. Therefore, other ice sheet modelling communities can benefit from the implemented glacier model coupling workflow template and couple their models with much less technical effort. Similarly, the workflow can be extended with new script tasks or control constructs (conditional or iterative composites) from any part of the workflow structure, according to the scenario requirements.

### 5.7 Resource management-agnostic access

UNICORE-based job submission does not require a hard-coded approach, as it provides users with a seamless interface for varying resource management systems (e.g. SLURM, LSF etc.). The underlying server-side UNICORE services automatically take care of the job submission commands' translation to the target batch system on behalf of the user. However, while this middleware abstraction offers user-friendliness, it also means that, due to the abstraction, some batch system-specific low-level technical features that might be useful for the application are truncated. This is because the middleware layer hides them for the sake of a unified, less complex and consistent user experience. If any batch system-specific features are necessary, workflow tasks could still be partially reprogrammed to use custom features. In our case study, we needed only standard job submission features, so no custom batch system-specific functions were required.

### 5.8 Reproducibility and reuse

As in every field of science, it is necessary that other scientists (and also the original scientist) are able to reproduce and validate the glaciological results that we obtained. However, it is not a trivial task to ensure this, in particular if the resource environment has changed, e.g. when using a different or updated HPC cluster or batch system.

In the UNICORE-based implementation (where the batch system translation is automated as described in Sect. 5.7 and hard-coded, system-dependent paths can be easily avoided), this scenario is much simplified for the end users, as for them just the UNICORE workflow needs to be exported into a reusable workflow (using an XML format). The exported version can easily be imported by any other UNICORE system. The only requirement is that the imported workflow tasks' resource requirements and the shared workflow variables have to be realigned to the new environment (which might, e.g. have a lower number of CPU cores). If the cluster environment stays the same, and only the user changes, there is no need to reconfigure target resource requirements, and only the shared workflow variables concerning the storage of user-specific data, e.g. the location of the data sets to be used as input, need to be adjusted. In addition to reproducing experiments, the high-level UNICORE workflows can also easily be extended (cf. Sect. 5.6) by other scientists from the glaciology community to adapt them to their needs.

## 6 Conclusions

Scientific workflows automate and enable complex scientific computational scenarios, which include data-intensive scenarios, parametric executions, and interactive simulations. In this article, a glacier ice flow and calving model have been combined into a single high-level scientific workflow. The ice flow was solved using Elmer/Ice, a glaciological extension to the finite element model (FEM) code Elmer, whereas calving was simulated by the discrete element code Helsinki Discrete Element Model (HiDEM).

We created a workflow implementation based on the state-of-the-art UNICORE middleware suite. The workflow can easily be composed on a high and abstract level through the UNICORE rich client's visual workflow editor. The workflow developed this way can be submitted asynchronously to HPC clusters, while real-time monitoring is also possible. Such a versatile GUI-type environment that allows for automated error reporting and handling clearly gives an operational advantage. It helps to reduce the time needed to debug set-ups prior to production and helps to deal more effectively with unexpected problems during the production phase. For this case study, the production deployment of UNICORE instances on CSC's *Taito* and *Sisu* clusters was used.

We evaluated our workflow implementation from different points of view, such as streamlining coupling-related I/O, improving CPU utilization, workflow extensions, usability, and portability in order to foster reproducibility. Unnecessary file copying was removed, which sped up I/O and, in turn, the whole workflow. The abstracted workflow allocated only as many CPU cores as needed for each step, thus avoiding idle cores and making them available for other jobs. The UNICORE-based workflow implementation can be exported to an abstract machine-readable format, so that other users interested in reproducing results can easily re-run the simulation without any change on a different platform and still obtain the same outputs generated by our experiment, or they can reuse the workflow to adapt and apply it to new data sets. Furthermore, this UNICORE-based workflow can easily be changed by simply updating the existing tasks, given that they are properly configured with respect to the target resources needed. The workflow can also be easily extended due to the inherent high-level of abstraction.

While we demonstrated the workflow management system approach based on the glacier modelling case study, we believe that the requirements that we derived from our case study are in fact applicable to many other Earth science modelling applications, and even to further scientific disciplines that involve distributed HPC resources. Hence, our UNICORE-based approach also appears promising for other case studies. The high-level workflow approach allows one to focus on critical workflow-related aspects such as optimized coupling-related I/O, improved CPU utilization, and reproducibility.

With regard to future work, it would be possible to create a portal that provides even simpler access to our glaciology workflow. This might include a web-based interface where a scientist simply has to upload their credentials for using an HPC cluster, upload or point to the data set to be used, configure some simulation-specific parameters, and finally press a button to start the workflow on one or multiple HPC clusters.

As the UNICORE workflow management system is independent from any scientific domain, we encourage other disciplines to transform and automate their application scenarios into automated workflows. We have already successfully demonstrated this for applications in remote sensing (Memon et al., 2018a) and for the interpretation of analytical ultracentrifugation experiments (Memon et al., 2013a).

*Code availability.* Elmer and Elmer/Ice sources are freely available at https://github.com/ElmerCSC/elmerfem (last access: 27 June 2019) – note our comments on scaling and performance improvements (Zwinger et al., 2013). The initial coupling (shell) scripts can be obtained by directly contacting the author. HiDEM sources can be openly accessed via https://doi.org/10.5281/zenodo.1252379 (Todd andÅström, 2018). All of the UNICORE packages are available at http://unicore.eu (last access: 25 June 2019). The UNICORE-based glacier model coupling workflow that can be used as template and the associated Bash helper scripts are available via https://doi.org/10.23728/b2share.f10fd88bcce240fb9c8c4149c130a0d5 (Memon et al., 2018b).

To access and run the workflow, the UNICORE sites and the workflow services, as well as the application packages and the coupling scripts have to be installed. The intended users need to have appropriate and valid X.509 credentials, as UNICORE rich client (URC) requires them to present the credentials when accessing the workflow services.

*Author contributions.* SM developed the overall workflow design, prototyping, development, and realization, which also includes the workflow deployment and administration. SM also streamlined the glacier modelling case study scripts and application deployment on CSC clusters. DV contributed the scientific problem that motivated the construction of this workflow. DV also developed the coupling script, and provided significant contributions during the workflow development, testing, and deployment phases. TZ is involved in the Elmer/Ice project and provided support during the integration and deployment of Elmer/Ice. JÅ is the main author of HiDEM. HN provided input on workflow foundations, conclusions, and also worked on streamlining the paper. In addition, he was in charge of the application to NeIC Dellingr for CPU time on CSC clusters. MR provided related work highlighting scientific workflows and Earth science modelling. MB contributed to the system architecture from a software engineering perspective.

*Competing interests.* The authors declare that they have no conflict of interest.

*Acknowledgements.* The authors wish to acknowledge CSC – IT Center for Science, Finland, for computational resources that were granted via the Nordic e-Infrastructure Collaboration (NeIC) Dellingr project. Furthermore, Jülich Supercomputing Centre, Germany, provided the development test bed for the workflow development and job management service deployments, and also hosted the central workflow management services that orchestrated and dispatched workflow tasks to the CSC computing sites. The authors are grateful to Jaakko Leinonen (CSC) for his support and patience in making the UNICORE services available on CSC's computing resources, and to Joe Todd (University of St. Andrews) for publishing HiDEM on GitHub.

*Financial support.* This work was supported by NordForsk as part of the Nordic Center of Excellence (NCoE) eSTICC (eScience Tools for Investigating Climate Change at High Northern Latitudes; grant no. 57001) and the NCoE SVALI (Stability and Variation of Arctic Land Ice; grant no. 24420).

The article processing charges for this open-access publication were covered by a Research Centre of the Helmholtz Association.

*Review statement.* This paper was edited by Alexander Robel and reviewed by two anonymous referees.

## References

Åström, J. A., Riikilä, T. I., Tallinen, T., Zwinger, T., Benn, D., Moore, J. C., and Timonen, J.: A particle based simulation model for glacier dynamics, The Cryosphere, 7, 1591–1602, https://doi.org/10.5194/tc-7-1591-2013, 2013.

Åström, J. A., Vallot, D., Schäfer, M., Welty, E. Z., O'Neel, S., Bartholomaus, T. C., Liu, Y., Riikilä, T. I., Zwinger, T., Timonen, J., and Moore, J. C.: Termini of calving glaciers as self-organized critical systems, Nat. Geosci., 7, 874–878, https://doi.org/10.1038/ngeo2290, 2014.

Barker, A. and van Hemert, J.: Scientific Workflow: A Survey and Research Directions, in: Parallel Processing and Applied Mathematics. PPAM 2007, Lecture Notes in Computer Science, Springer, https://doi.org/10.1007/978-3-540-68111-3_78, 2008.

Bassis, J. and Jacobs, S.: Diverse calving patterns linked to glacier geometry, Nat. Geosci., 6, 833—836, https://doi.org/10.1038/ngeo1887, 2013.

Cuffey, K. M. and Paterson, W. S. B.: The physics of glaciers, Academic Press, Cambridge, MA, USA, 2010.

Dagum, L. and Menon, R.: OpenMP: an industry standard API for shared-memory programming, IEEE Comput. Sci. Eng., 5, 46–55, https://doi.org/10.1109/99.660313, 1998.

Deelman, E., Vahi, K., Juve, G., Rynge, M., Callaghan, S., Maechling, P. J., Mayani, R., Chen, W., Ferreira da Silva, R., Livny, M., and Wenger, K.: Pegasus: a Workflow Management System for Science Automation, Future Gener. Comp. Sy., 46, 17–35, https://doi.org/10.1016/j.future.2014.10.008, 2015.

Demuth, B., Schuller, B., Holl, S., Daivandy, J., Giesler, A., Huber, V., and Sild, S.: The UNICORE Rich Client: Facilitating the Automated Execution of Scientific Workflows, 2013 IEEE 9th International Conference on e-Science, 23–25 October 2013, Beijing, China, 238–245, https://doi.org/10.1109/eScience.2010.42, 2010.

Eclipse Foundation: Eclipse Foundation: Rich Client Platform (RCP), available at: https://wiki.eclipse.org/Rich_Client_Platform (last access: 23 May 2018), 2013.

Ferreira da Silva, R., Filgueira, R., Pietri, I., Jiang, M., Sakellariou, R., and Deelman, E.: A Characterization of Workflow Management Systems for Extreme-Scale Applications, Future Gener. Comp. Sy., 75, 228–238, https://doi.org/10.1016/j.future.2017.02.026, 2017.

Frey, J.: Condor DAGMan: Handling Inter-Job Dependencies, presentation slides, available at: http://www.bo.infn.it/calcolo/condor/dagman/ (last access: 25 May 2018), 2003.

Gagliardini, O., Zwinger, T., Gillet-Chaulet, F., Durand, G., Favier, L., de Fleurian, B., Greve, R., Malinen, M., Martín, C., Råback, P., Ruokolainen, J., Sacchettini, M., Schäfer, M., Seddik, H., and Thies, J.: Capabilities and performance of Elmer/Ice, a new-generation ice sheet model, Geosci. Model Dev., 6, 1299–1318, https://doi.org/10.5194/gmd-6-1299-2013, 2013.

Geuzaine, C. and Remacle, J.-F.: Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities, Int. J. Numer. Meth. Eng., 79, 1309–1331, https://doi.org/10.1002/nme.2579, 2009.

Greve, R. and Blatter, H.: Dynamics of ice sheets and glaciers, Advances in Geophysics and Environmental Mechanics and Mathematics, Springer, Heidelberg, Germany, 2009.

Kääb, A., Lefauconnier, B., and Melvold, K.: Flow field of Kronebreen, Svalbard, using repeated Landsat 7 and ASTER data, Ann. Glaciol., 42, 7–13, https://doi.org/10.3189/172756405781812916, 2005.

Luckman, A., Benn, D. I., Cottier, F., Bevan, S., Nilsen, F., and Inall, M.: Calving rates at tidewater glaciers vary strongly with ocean temperature, Nat. Commun., 6, 8566, https://doi.org/10.1038/ncomms9566, 2015.

Memon, M. S., Memon, A. S., Riedel, M., Schuller, B., Mallmann, D., Tweddell, B., Streit, A., van den Berghe, S., Snelling, D., Li, V., Marzolla, M., and Andreetto, P.: Enhanced Resource Management Capabilities using Standardized Job Management and Data Access Interfaces within UNICORE Grids, in: International Conference on Parallel and Distributed Systems (ICPADS), 5–7 December 2007, Hsinchu, Taiwan, IEEE, https://doi.org/10.1109/ICPADS.2007.4447834, 2007.

Memon, S., Attig, N., Gorbet, G., Gunathilake, L., Riedel, M., Lippert, T., Marru, S., Grimshaw, A., Janetzko, F., Demeler, B., and Singh, R.: Improvements of the UltraScan scientific gateway to enable computational jobs on large-scale and open-standards based cyberinfrastructures, in: Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery, XSEDE '13, 22–25 July 2013, San Diego, CA, USA, 39:1–39:7, ACM, New York, NY, USA, https://doi.org/10.1145/2484762.2484800, 2013a.

Memon, S., Holl, S., Schuller, B., Riedel, M., and Grimshaw, A.: Enhancing the Performance of Scientific Workflow Execution in e-Science Environments by Harnessing the Standards Based Parameter Sweep Model, in: Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery, XSEDE '13, 22–25 July 2013, San Diego, CA, USA, 56:1–56:7, ACM, New York, NY, USA, https://doi.org/10.1145/2484762.2484820, 2013b.

Memon, S., Cavallaro, G., Hagemeier, B., Riedel, M., and Neukirchen, H.: Automated Analysis of remotely sensed images using the UNICORE workflow management system, in: 2018 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), 23–27 July 2018, Valencia, Spain, IEEE, https://doi.org/10.1109/IGARSS.2018.8519364, 2018a.

Memon, S., Vallot, D., Riedel, M., Neukirchen, H., Zwinger, T., and Book, M.: Elmer-HiDEM Workflow UNICORE, https://doi.org/10.23728/b2share.f10fd88bcce240fb9c8c4149c130a0d5, 2018b.

Message Passing Interface Forum: MPI: A Message-Passing Interface Standard, Version 3.0, University of Tennessee, Knoxville, TN, USA, available at: https://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf (last access: 25 June 2019), 2012.

Nuth, C., Schuler, T. V., Kohler, J., Altena, B., and Hagen, J. O.: Estimating the long-term calving flux of Kronebreen, Svalbard, from geodetic elevation changes and mass-balance modelling, J. Glaciol., 58, 119–133, https://doi.org/10.3189/2012JoG11J036, 2012.

Råback, P.: ElmerGrid Manual, CSC – IT Center for Science, available at: http://nic.funet.fi/index/elmer/doc/ElmerGridManual.pdf (last access: 20 June 2018), 2015.

Råback, P., Malinen, M., Ruokolainen, J., Pursula, A., and Zwinger, T.: Elmer Models Manual, CSC – IT Center for Science Ltd., Espoo, available at: http://www.nic.funet.fi/index/elmer/doc/ElmerModelsManual.pdf, last access: 20 June 2018.

Schellenberger, T., Dunse, T., Kääb, A., Kohler, J., and Reijmer, C. H.: Surface speed and frontal ablation of Kronebreen and Kongsbreen, NW Svalbard, from SAR offset tracking, The Cryosphere, 9, 2339–2355, https://doi.org/10.5194/tc-9-2339-2015, 2015.

Streit, A., Bala, P., Beck-Ratzka, A., Benedyczak, K., Bergmann, S., Breu, R., Daivandy, J. M., Demuth, B., Eifer, A., Giesler, A., Hagemeier, B., Holl, S., Huber, V., Lamla, N., Mallmann, D., Memon, A. S., Memon, M. S., Rambadt, M., Riedel, M., Romberg, M., Schuller, B., Schlauch, T., Schreiber, A., Soddemann, T., and Ziegler, W.: UNICORE 6 – Recent and Future Advancements, Ann. Telecommun., 65, 757–762, https://doi.org/10.1007/s12243-010-0195-x, 2010.

Todd, J. and Åström, J.: HiDEM, Zenodo, https://doi.org/10.5281/zenodo.1252379, 2018.

Vallot, D., Pettersson, R., Luckman, A., Benn, D. I., Zwinger, T., Van Pelt, W. J. J., Kohler, J., Schäfer, M., Claremar, B., and Hulton, N. R. J.: Basal dynamics of Kronebreen, a fast-flowing tidewater glacier in Svalbard: non-local spatio-temporal response to water input, J. Glaciol., 63, 1012—1024, https://doi.org/10.1017/jog.2017.69, 2017.

Vallot, D., Åström, J., Zwinger, T., Pettersson, R., Everett, A., Benn, D. I., Luckman, A., van Pelt, W. J. J., Nick, F., and Kohler, J.: Effects of undercutting and sliding on calving: a global approach applied to Kronebreen, Svalbard, The Cryosphere, 12, 609–625, https://doi.org/10.5194/tc-12-609-2018, 2018.

van Pelt, W. and Kohler, J.: Modelling the long-term mass balance and firn evolution of glaciers around Kongsfjorden, Svalbard, J. Glaciol., 61, 731–744, https://doi.org/10.3189/2015JoG14J223, 2015.

Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., Soiland-Reyes, S., Dunlop, I., Nenadic, A., Fisher, P., Bhagat, J., Belhajjame, K., Bacall, F., Hardisty, A., Nieva de la Hidalga, A., Balcazar Vargas, M. P., Sufi, S., and Goble, C.: The Taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud, Nucleic Acids Res., 41, 557–561, https://doi.org/10.1093/nar/gkt328, 2013.

Zwinger, T., Malinen, M., Ruokolainen, J., and Råback, P.: Scaling and Performance Improvements in Elmer/Ice, Zenodo, https://doi.org/10.5281/zenodo.822189, 2013.