CloudFlow - An Infrastructure for Engineering Workflows in the Cloud

Håvard Heitlo Holm*, Jon M. Hjelmervik* and Volkan Gezer†
*Heterogeneous Computing Group
SINTEF ICT
Oslo, Norway
Emails: havard.heitlo.holm@sintef.no
and jon.m.hjelmervik@sintef.no
†Innovative Factory Systems (IFS)
German Research Center for Artificial Intelligence (DFKI)
Kaiserslautern, Germany
Email: volkan.gezer@dfki.de

Abstract—**In this paper, we present a framework for easy integration of existing software solutions in a cloud environment. We aim to allow software providers to offer their products in the Cloud through a common web portal. Central in the framework is workflows, where independent software solutions can be chained together to solve the users tasks in a unified way. One of the main challenges addressed here, is facilitating workflows spanning multiple software vendors and cloud solutions. The framework is validated by a range of experiments, consisting of both software vendors and end users, from the context of manufacturing industries, and offer unique and ubiquitous access to integrated workflows.**

Keywords–Workflows; Cloud computing; HPC; Semantic descriptions; One-stop-shop.

I. Introduction

Cloud computing is now a natural part of the daily life, both professionally and for consumers. Users are expecting to have access to all their software and data independent of which computer they are using, which have revolutionized how data is consumed and shared.

Cloud providers are continuously extending and improving their tools to make it easier to deploy software in their solutions. These developments have made cloud computing attractive for software vendors, and many companies offer cloud integration throughout their product range. However, it can also be very tempting to make the cloud integration very tight, which in turn locks the software vendor into the cloud provider's ecosystem. New solutions, including UberCloud [1] and Cloud Modelling Language (CloudML) [2], target this challenge by offering platforms that are neutral to cloud providers, while aiming at making it as easy as possible to offer cloud based solutions for their costumers' software.

Not all tasks can be completed using a single application, but are best solved using a workflow where data is transferred from one application to another. The ideal workflow may consist of software from different vendors, which should seamlessly work together. To achieve this, it is not sufficient to give access to software in a cloud solution, the interoperability between the different applications and software suits must also be targeted. This paper proposes to add semantic descriptions to the available software, as well as their input and outputs. The semantic descriptions are used to chain compatible services into complete workflows. Furthermore, data formats should be based on open standards, or come with conversion tools to and from standard formats. In this paper, we present the CloudFlow Infrastructure, aiming at providing the technology platform for a one-stop-shop for cloud based workflows. This work builds upon the initial results by Stahl et al. [3].

The infrastructure described here can be applied to any business area, however, data formats and descriptions must be adapted to the application domain. However, currently the focus is on manufacturing industries, with a special focus on the needs of small businesses. The examples used in this paper therefore come from this domain, though the technology is neutral to application domain. In manufacturing industries, different software suits are used across the lifetime of their products, from design through numerical analysis through quality assurance and maintenance. Small companies in this market often find it too expensive to install the different solutions locally, due to hardware costs, installation overhead and license costs. This may cause loss in quality of their products due to insufficient analysis, and overly expensive design phases due to inefficient work procedures. For such companies, having access to a cloud solution that spans over different clouds and software providers, all integrated in tailored workflows, will not only save time and cost, but also improve their final product.

Larger companies may already have a server infrastructure and prefer to store their data within their control. These companies may still benefit from the integrated workflows and access from everywhere inside the secure network. It is therefore important that private cloud solutions also are supported to host the CloudFlow Infrastructure.

End users that are familiar with a software solution may be reluctant to shift providers when moving from a desktop application to a cloud based approach. The platform must therefore not only be attractive for end users, but also for software providers. This means that it must be flexible enough to support the wide range of software solutions desired and have a good user experience, while keeping the costs low.

The goal is to provide ubiquitous availability of compute resources, software and data. In contrast to other approaches, the aim here is to integrate existing software solutions into one common platform, combining them to work together and make them accessible through a web portal. Furthermore, the proposed solution supports installation in private clouds as well as access to multiple cloud and High Performance Computing (HPC) providers through one common portal installation. To facilitate a broad selection of existing software solutions, we target cloud solutions offering Infrastructure as a Service, where software providers can install their own operating system and fully control the virtual machines.

In the rest of this paper, we give a short overview of the related work in Section II. The CloudFlow Infrastructure is then presented in Section III, where the focus is on the aspects and infrastructure components related to workflow orchestration and execution, resource monitoring, data storage, authentication, and utilization of HPC clusters. In order to validate the infrastructure and demonstrate the usefulness of CloudFlow, the experimental setup applied through the CloudFlow project is described in Section IV, along with an example workflow. Finally, we conclude and describe some plans for future work in Section V.

II. Related Work

Cloud computing and cloud-based engineering is delivered by multiple providers. One dedicated software vendor delivering such a solution is SimScale [4], offering simulators for computational fluid dynamics, finite element analysis and thermodynamics in the cloud. Based on these simulation tools and web-based visual pre- and post-processing, Simscale targets end users only. Combining their cloud solution with software developed by other vendors is therefore not straight forward.

The cloudSME project [5] combines a business model targeting both end users and software vendors. Software vendors are offered a Platform as a Service solution, where they offer Software as a Service for their existing and new end users. This approach also makes it possible for end users to combine software from different software vendors to perform more complex engineering tasks. CloudSME does however not use any semantic information to orchestrate how to combine the different software, and lacks the use of HPC.

There are several initiatives to simplify cloud deployment and avoid vendor lock-in. Bergmayr et al. [2] propose a modelling approach, where the cloud deployment is described by a vendor independent language CloudML. The deployment model is implemented in this language, which sets up virtual machines, network communication and deploys services accordingly.

Stahl et al. [3] proposed the initial work and the main concepts of the CloudFlow Infrastructure. Among the newly introduced concepts are a unified way to access HPC resources, functionality to use external cloud providers, resource monitoring and a graphical tool to define workflows.

Semantic Markup for Web Services (OWL-S) and *Business Process Execution Language* (BPEL) are two technologies that allow web service execution as processes. BPEL is a language to execute business processes with web services as stated by Mendling [6]. According to its specifications, BPEL executes web services defined using Web Service Description Language (WSDL). It supports orchestration of actions within services, by structuring them as sequences and supporting branches and loops. The structure is described using a syntax based on Extensible Markup Language (XML). OWL-S is a markup that is built on top of Web Ontology Language (OWL) and describes web services semantically introducing an XML-based syntax. It also supports orchestration and due to semantic technologies, structuring the sequences using OWL-S is both machine and human understandable. OWL-S and WSDL are usually used to describe services based on the Simple Object Access Protocol (SOAP) specification. It allows web services to send requests in a predefined structure encoded in a XML format.

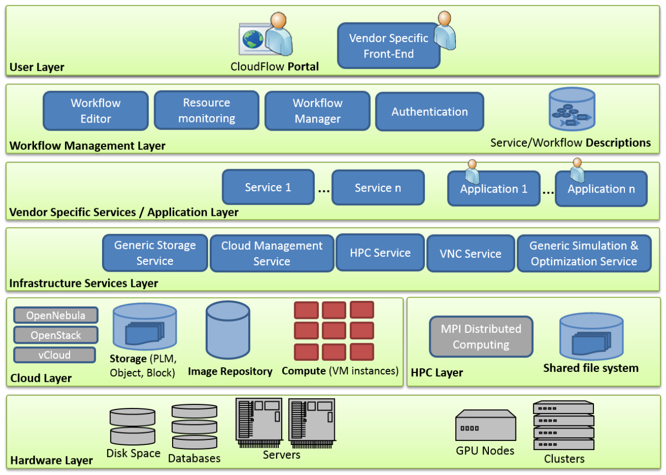

Figure 1. Simplified diagram of the system layers with their main components.

BPEL is similar to OWL-S in terms of orchestration and XML-based syntax, but it lacks utilizing semantic technologies. Therefore, making web services machine-understandable and automating them without user interaction is a non-trivial task using BPEL [7].

The process execution is usually performed by designing an execution order and monitoring it with an execution engine. There are several execution engines designed for this purpose. Execution engines, or managers, introduce an editor or a syntax to specify the order and then track the progress. Depending on the implementations, they can also provide user interface.

One of the available execution engines for BPEL is Process Manager by Oracle. It provides a graphical interface to manage cross-application business processes in a service oriented architecture (SOA) [8]. It also allows designing workflow steps and connecting external systems into the workflow. However, lack of semantic technologies inside BPEL prevents automation of these design steps. Involving semantic technologies would therefore increase the productivity by reducing the time needed to design the task steps, and is hence quite important.

To achieve the goal of CloudFlow, a manager which can facilitate semantic technologies, integrate web services from different providers and locations, and provide automation during the design and execution phase was necessary.

III. Architecture Overview

The components in the CloudFlow Infrastructure can be described as a multilayer architecture to separate functionalities. It consists of six main layers as depicted in Fig. 1. The natural entry point for both end users and software vendors is through the CloudFlow Portal, or just *the Portal*, found in the user layer of the infrastructure. This section presents key components and concepts of the CloudFlow Infrastructure.

A. Workflow Management

A workflow is an orchestrated and repeatable pattern of several activities enabled by the systematic organization of resources into processes that transform materials, provide services, or process information. Workflows may be as trivial as browsing a file structure or visualizing a Computer-Aided Design (CAD) model, or they can be more complex, including describing the full set of operations used to design, analyse and prepare for manufacturing of a product. Utilizing semantic technologies such as OWL-S makes it possible to reuse existing workflows as building blocks in more complex ones.

As described in Section I, the main goal of CloudFlow is to host software from different software providers and chain appropriate parts of them to perform end user tasks in workflows within one common platform. In the following, we will describe how web services are integrated into the CloudFlow Infrastructure, and how workflows are designed and executed.

1) Services: A set of complementary reusable functionalities that are provided by a software for different purposes is called "service." More particularly, a web service is a software system designed to support interoperable machine-to-machine interaction over a network [9]. A web service invocation consists of a single request/response pair and is expected to execute in a short time.

The CloudFlow Infrastructure defines some Application Programming Interface (API) requirements to services to be integrated into workflows. The simplest web service that follows those requirements is called a *synchronous service*. These services are useful when the operation ends relatively quickly, and the user does not need progress update during its execution. In contrast, *asynchronous services* do not have any restrictions when it comes to runtime, and they are also allowed to present the user with progress information. Common for the service types are that they represent an operation that takes predefined input parameters and generates output parameters without a user interaction. All communication with the services are performed through SOAP requests which are sent to their endpoints defined through their WSDL files. In addition to services, also *applications* which allow user interaction are supported. The most common application types are web forms for selecting parameters for the following service and visualization applications for inspecting the output. Throughout this paper, the term *CloudFlow service* denotes services and applications compatible with the CloudFlow API.

2) Workflow Definition: In order to define a web service as a CloudFlow service, and then to create workflows from a chain of CloudFlow services, the web services need to be integrated into the infrastructure. This is performed using a graphical tool named *Workflow Editor*.

Workflow Editor (WFE) was mentioned as future work in [3] as an implementation of a workflow modelling tool and it is currently available for use. The developed WFE is based on XML, SOAP, and WSDL standards, and inserts web services into a semantic database by adding semantic descriptions based on their WSDL. The server side functionality of WFE is implemented as web services hence it also provides a SOAP API that can be used directly from 3rd party tools.

In order to make a web service available as a CloudFlow service, the service provider supplies the endpoint for the service's WSDL to WFE, where semantic descriptions of the service itself and its input and output parameters are defined. Based on these semantic descriptions, semi-automatic orchestration of workflows can be made by letting the system suggest services which are compatible with respect to their inputs and outputs. After these parameters are converted into semantic descriptions for the given service, the service is integrated into the infrastructure and can be used as a step in a workflow.

The WFE is also used to chain services into workflows. The creation is offered through both a textual and a graphical editor, combined as a single web page. Using the graphical editor, the CloudFlow services are selected using a dropdown menu and appended to a workflow. The data flow between the services is defined by dragging and dropping outputs of services and connecting them with inputs of others. The execution order is represented using dotted arrows and the data flow is shown using solid lines as shown in Fig. 2. Each action performed using the graphical editor is synchronized with an XML-based meta-formatted textual editor. The XML format contains all information stored in the semantic database, and can be used also for later updates of the workflow.

Even though each CloudFlow service typically represents an individual operation with dedicated input and output parameters, some services naturally belong together. Instead of having to connect the same sequence of services repeatedly for multiple different workflows, such services are modelled as a smaller workflow of their own called *sub-workflows*. Sub-workflows are yet again available to be added into any other workflow as a single component, similar to a regular CloudFlow service. The changes made within a sub-workflow are applied to all workflows using it, reducing time and effort for the workflow designer through avoiding a repetitive task.

3) Workflow Manager: The semantic descriptions created by WFE only contain meta-data and describe how the data is bound. The component *Workflow Manager* (WFM) acts as an execution engine acting on the semantic descriptions. It executes and monitors all services in a workflow providing the input parameters as defined in WFE, either as constant values or outputs from previous steps. The status of each asynchronous service is checked at regular intervals in order to determine if it is finished as well as to present the service's status to the user.

Before executing a workflow, the billing component is asked to verify that the user has the valid licenses for all involved services. Therefore, requests to initiate services that do not originate from the WFM should be rejected, preferably on a network level, e.g., by strict firewall settings. The WFM also tracks execution times by utilizing resource monitoring components which are explained in Section III-B. Usually, the user interacts with the Portal to initiate and monitor workflows. However, this is just one possible way and the Portal is not needed during the execution of workflows. Both synchronous and asynchronous services can be executed even after the user has left the client that initiated the workflow. This is also true for applications, though they will wait for user interaction before the workflow can be continued. Independently on how a workflow is initiated, the Portal can be used to inspect the status of workflows and interact with them during their execution.

B. Resource Monitoring and Billing

To facilitate that CloudFlow becomes a one-stop-shop where software vendors integrate and offer their software for new customers, functionality to monitor the resource usage by each workflow and service is needed. Based on different requirements, the software vendors are able to use different business models, such as offering their software as pay-per-use, or for a fixed monthly or annual fee. For computationally intensive software, that require exclusive access to a hardware resource, there will also be a cost related to the CPU hours

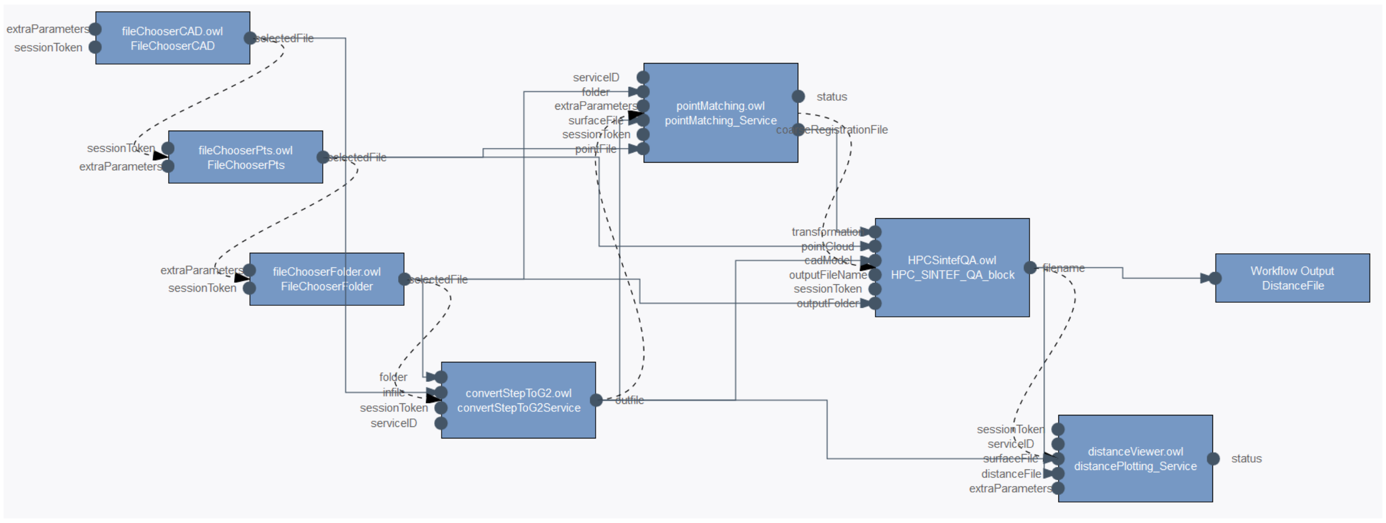

Figure 2. The graphical user interface of WFE, where dotted lines represent flow of services, while solid lines connect parameters from outputs to inputs.

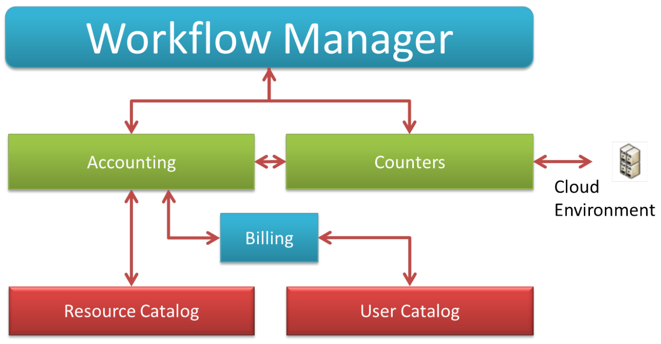

Figure 3. Communication between billing-related components inside infrastructure.

spent. Each service is tagged with which *software package* it belongs to. All license costs related to the software package is described in the *Resource Catalog* component, either as a time based license requirement, runtime cost, or a combination of both. The Resource Catalog also holds information about hardware costs and which vendors the costs belong to. Fig. 3 illustrates how the different components are connected.

As a workflow is executed, the resources consumed through this workflow is gathered, and the related cost for running the workflow is calculated. Collection of this information and cost calculation is performed within the resource monitoring and billing components of the CloudFlow Infrastructure. When an end-user starts a workflow, the WFM lists all software packages within the workflow and checks with *Accounting* service whether the user has the required licences to run them. Later, if the user is allowed to execute the workflow, the *Counter* service is used to track the execution times delivered by the WFM. This service passes these data back to the Accounting service to calculate the bill for CPU usage, as well as the software costs within the workflow.

The Resource Catalog holds a list of software and hardware/data centers, whereas the *User Catalog* keeps a list of organizations or users. The calculations and usage information at the end of each workflow are gathered by the *Billing* component. This component issues bills to the end users and is the only component which users interact with in order to get their usage and cost reports.

C. Cloud Storage

In order to follow the loosely-coupled layered architecture design of the CloudFlow Infrastructure (see Fig. 1), the interaction with the cloud storage is designed to be vendor independent. As different cloud storage solutions have different APIs for accessing files, a set of services with a unified API is required. Further, in order to avoid unnecessary network cost and to avoid potential security issues, the files need to be transferred directly between the cloud storage and the client, and not via an intermediate server. To support these requirements, the *Generic Storage Services* (GSS) have been developed within CloudFlow.

The GSS exposes an API consisting of both SOAP and REpresentational State Transfer (REST) web services, and offers functionality for interacting with the cloud storage solutions available in CloudFlow. In contrast to SOAP, RESTful web services come with a smaller overhead and are better suited for transferring large amounts of data. Each available storage solution is added as a back-end to GSS, where information is provided through SOAP services on how to use the native REST interface. The client transfers files directly to and from the cloud storage, with no added overhead. In this way, GSS acts like a look up service telling how to make requests toward the different back-ends, where each back-end is treated as an object storage. Beside transferring files, other functionality such as listing folder content, checking existence of files, creating folders etc., are made directly through the SOAP API.

Files are uniquely defined by file IDs, which includes a prefix indicating which storage back-end the file belongs to. The file ID combined with a valid authentication token is sufficient for downloading any file. As long as all CloudFlow services are implemented using this API, interoperability and vendor independent file access is obtained within CloudFlow workflows. Further, any cloud storage solution with a RESTful API can be made available in CloudFlow by adding an additional back-end in GSS. Existing services can then immediately use the new cloud storage solution without making any changes to their implementation. A cloud storage can also use external authentication solutions, as it is not a requirement that the authentication token used towards the cloud storage is a valid CloudFlow token.

A web-based file browser application is available for all workflows. It is configurable through WFE to tailor its behaviour for each workflow, and has a user friendly interface with a context dependent right-click menu. Here, end users can upload and download files between their computers and the cloud.

Since the CloudFlow services have SOAP interfaces, their parameters should consist of short messages rather than entire files. Because of this, the file browser gives the file ID of the chosen files as output instead of the content of it. This rule illustrate best practice and applies to all CloudFlow services.

Currently, four cloud storage solutions are made available in the CloudFlow Infrastructure through GSS. There are two OpenStack Swift installations (one internal and one external), a product lifecycle management (PLM) system, and a native storage at one of CloudFlow's HPC providers.

D. Authentication and Multi-Cloud

Several of the CloudFlow Infrastructure components need to be available from outside the infrastructure itself. This includes design and execution of workflows, interaction with the cloud storage, and viewing how much resources a user has spent. Since only registered users should be allowed to issue such requests, and since a user should only have access to their own files and resource usage, all web services within the CloudFlow Infrastructure need an authentication parameter representing the user. For this, a token based authentication system is used. Users obtain a token at login which represents them throughout the session, and which contains their appropriate permissions.

As a security measure, tokens have limited life spans, meaning that requests containing old tokens will be unsuccessful. However, workflows (and perhaps especially within manufacturing industries) can consist of services lasting longer than any lifespan given to tokens. Since an expired token can not be used for, e.g., uploading results to the cloud storage, CloudFlow needs an authentication scheme that both invalidates tokens after a certain time while allowing workflows of arbitrary lengths to have access to infrastructure components.

In order to support these requirements, the *Authentication services* are introduced to CloudFlow. These services build on top of OpenStack's Keystone component [10], and extend its with functionality to handle the challenge related to long lasting workflows. In addition, vendor lock-in towards OpenStack is avoided through these services. Changing the communication with Keystone, or exchanging Keystone itself, will require changes in the implementation of the Authentication services only, while the API, and all components relying on the authentication, are kept unchanged.

The problem consisting of tokens expiring during workflow executions are solved by issuing and storing special workflow tokens from the Authentication services. Each time a workflow is started, such a token is created from combining the regular token with the ID of the workflow execution. This workflow token is stored in a database, and is passed to all services within the workflow. During validation, the regular

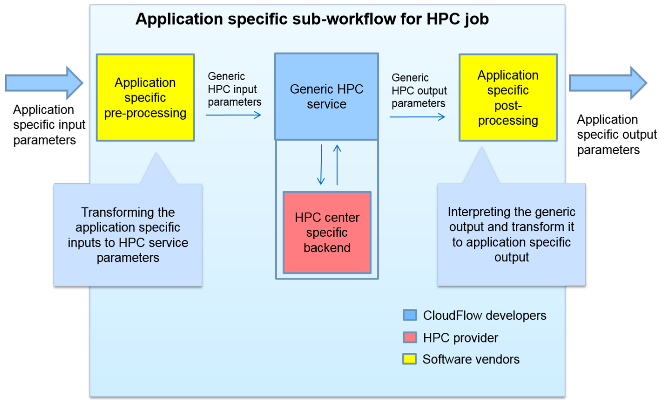
Application specific sub-workflow for HPC job
Application specific input parameters
Application specific pre-processing
Generic HPC input parameters
Generic HPC service
Generic HPC output parameters
Application specific post-processing
Application specific output parameters
HPC center specific backend
Transforming the application specific inputs to HPC service parameters
Interpreting the generic output and transform it to application specific output
CloudFlow developers
HPC provider
Software vendors

Figure 4. An application specific sub-workflow encapsulating an HPC job.

token is checked normally first, but if it has expired and the combination with the execution ID is recognized, the token is still marked as valid. A new regular and valid token can then be generated with the same permissions as the existing token based on the now invalid one. When the workflow later is finished or aborted, the special workflow token is deleted from the database, and is then invalidated.

1) Multi-Cloud: As CloudFlow is not tied to any one cloud, it is possible to use multiple clouds for hosting CloudFlow services. One reason for doing this might be that customers are physically too far away from the main CloudFlow cloud, making a local cloud more attractive in terms of network costs and delays. Other reasons might be that alternative clouds might be cheaper, or equipped with hardware not available in the CloudFlow cloud, for example by offering more powerful processing resources.

The main challenge related to such solutions is authentication across the different clouds. While the external clouds have their authentication methods, they are not necessarily compatible with those used in CloudFlow. Services that are written for multi-cloud settings should therefore be implemented with an additional external token parameter. The semantic information can then describe what cloud the external token should be authenticated against, and external authentication services can be added to such workflows. Such a service will provide a web form where the user can login to the external cloud, and where the external token is passed to the next steps in the workflow. The external token can also be stored in a cookie in the browser, so that if a valid token is already present, this token will be used and the users are spared from typing their username and password more than necessary.

E. HPC Access

Computationally intensive tasks that benefit from running on HPC clusters are common in many engineering workflows. Because of this, the CloudFlow Infrastructure is required to facilitate seamless and secure integration of such tasks, making it easy for software vendors to run their applications on an HPC cluster as part of a CloudFlow workflow. The solution for this is the design and concept of *HPC application sub-workflows*, with the generic *HPC service* as the central component.

An HPC application sub-workflow is built from three CloudFlow services as shown in Fig. 4; an application specific pre-processing service, the generic HPC service, and an application specific post-processing service. The HPC service is designed to be generic with respect to both the application and the type of job scheduling system used by the HPC provider. This way, the HPC providers can make changes to their queuing systems, or CloudFlow can expand to more HPC centers without requiring the software vendors to make changes to their services or workflows. Even though the HPC service is generic, the application that is executed through it will be part of a software package (as mentioned in Section III-B). The name of the software package it is part of will therefore be hardcoded as input to the HPC service within the application sub-workflow. This information is then used by the service to check with the Resource monitoring component that the user has a license to run the application, and to ensure that the software vendor receives the correct license fees. Other resource management tasks, such as reporting CPU hours spent on the computation, are also reported from within the generic HPC service.

In order to separate the service from the HPC center specific details, the HPC service communicates internally with an HPC back-end. Since user credentials defined in CloudFlow are not necessarily compatible with the user definitions in the HPC cluster, the back-end may perform a mapping between the two sets of user definitions through a method seen fit by the HPC provider. For security reasons, different CloudFlow users should not be assigned the same user on the cluster. The two HPC providers within CloudFlow currently use different approaches to solve this challenge. One provider has set up a pool of HPC users reserved for CloudFlow, where each execution of the HPC service is assigned an arbitrary user from the pool, which is then reserved until that execution has completed. To ensure security, the home directory of each such HPC user is deleted between executions of jobs. The other provider assigns a one-to-one mapping between the two types of users, so that a CloudFlow user is assigned a dedicated HPC user. This second approach is particularly suitable to private cloud installations of CloudFlow, where the same system administrators control both the cloud and compute cluster environments.

The input and output parameters for the HPC service is highly generic, and should support the vast majority of applications that will be run on the compute cluster. As input, the service takes a string containing the set of command lines that will execute the application with its correct input parameters. Additional inputs to the service are the number of cores and nodes to use, as well as a maximum execution time used to limit cost and stop non-converging simulations.

The output from the HPC service is a string which is read from a *result file*, written by the application. As this output parameter will be sent in a SOAP message, output files should not be written in the result file, but rather uploaded to the cloud storage. The file ID of the uploaded file should instead be written as the result.

During the execution of the application, the end user will often appreciate some feedback in the form of a progress report. What kind of progress report that makes sense to provide highly depends on the application, as some applications are only able to provide a progress bar, while others might create a status report including images and text. A reserved *status file* is being monitored by the HPC service, and any content of this file is interpreted as HTML and displayed in the browser for the user. The software vendor can therefore fill this file with any meaningful information seen fit, either directly from the application, or through a background process parsing the log file of the application into the status file.

In order to create the string with the set of command lines, an application specific pre-processing service is required in front of the HPC service. This service is implemented by the software vendor providing the application itself, and takes the same input parameters as the application. Similarly, in order to interpret the output from the HPC service and add semantic descriptions to it, an application specific post-processing service is called. It parses the output from the HPC service, which is the information written in the result file, into application specific output parameters. Since both these operations in most cases are basic string manipulations which finish immediately, both the pre- and post-processing services are usually implemented as synchronous services. In some cases however, it is natural to let the end-user choose how many nodes the application should use. It is then natural to either implement the pre-processing service as a web application instead, or have a web application where this choice is made before the pre-processing.

When the number of nodes is hardcoded within the application specific sub-workflow, all details concerning the HPC is hidden from whoever uses the sub-workflow in a larger workflow. The sub-workflow will have nothing but application specific inputs and outputs, and will therefore have the same interface in WFE as a single CloudFlow service running the same application in the cloud environment. The usage of the compute resources is therefore transparent for the user, and does not require that the end user (or workflow creator) knows the difference between cloud and HPC environments.

IV. Results

The development of the CloudFlow Infrastructure is organized to meet the requirements from end users in manufacturing industries and their software providers. This has given opportunity to validate our choices and arrange validations, where the end users test the platform and the deployed software.

A. Validation results

To facilitate the development and validation, three *waves of experiments* have been set up. In the first wave of experiments, all workflows were tailored towards the needs of hydropower engineers. Software from 6 different independent software vendors (ISVs) were integrated with the infrastructure and accessible through the cloud solution, and validated with one common end user. For the second and third wave, European software vendors and end users were invited to test the infrastructure and develop new workflows based on the needs of the end user. In total 14 new experiments were selected, each with one new end user.

The motivation to use the CloudFlow Infrastructure varies among the experiments, including attracting new customers, reducing license cost, reducing time spent to create a new product and improving the design of new products. The overall common goal is to enhance availability of easy to use software and computational resources through

- User friendly interfaces
- Easy access to cloud computing resources

For each experiment, the user requirement group of the project helped the end user to define usability criteria requirements. In parallel, the software vendors developed business plans for

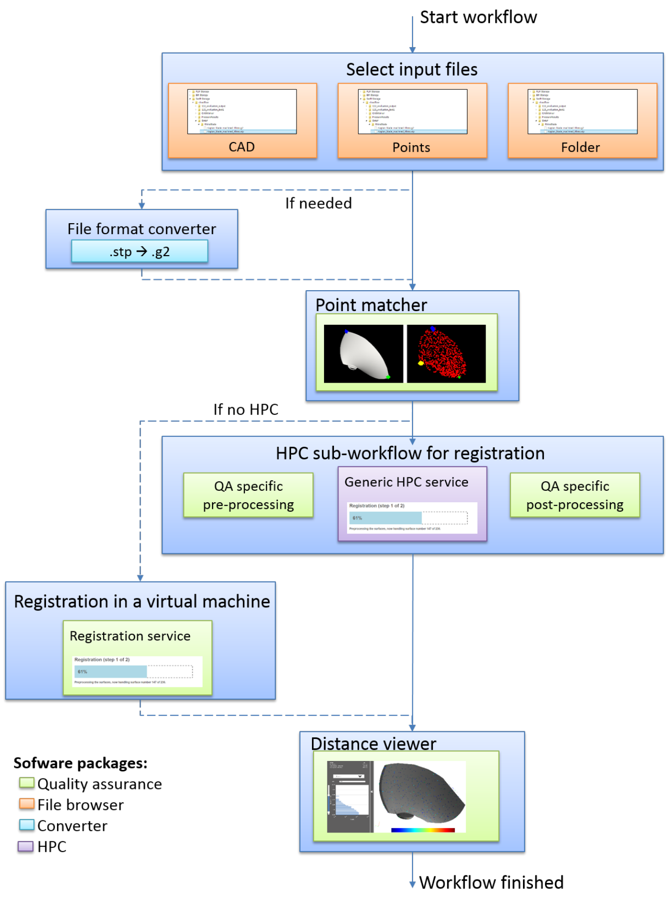

Figure 5. A workflow for quality assurance where a 3D scan of a produced product is compared with the initial CAD model.

how to realize the economical potential benefiting both the end user and software vendors. This way, not only the theoretical potential of the software platform is verified, but also that the final solution can sustain as an attractive option.

At several stages the end users were invited to perform daily tasks using the CloudFlow Infrastructure, validating the success, not only of the experiment but also the infrastructure as a whole.

The end users have demonstrated how to use the software solutions in a daily task. The demonstrations are monitored by the user requirements group to assess the user experience and recommend future improvements. The conclusion from the 13 already finished experiments have been processed and shows that the main goals of all experiments have been reached. Furthermore, the experiments reported that the end users would benefit economically from using more compute resources than today, and that the CloudFlow Infrastructure can make it economically feasible. The validation revealed that the concept is functional and makes it more attractive to use Cloud computing both for software vendors, consultancy companies and end users.

B. Quality Assurance in the Cloud

As an example of a workflow using the components and concepts mentioned in Section III, a quality assurance application will be considered. This workflow will compare a CAD model to a 3D scan of the produced product, for this application: a turbine blade. The goal is to confirm that the turbine blade is manufactured according to its design within given tolerances, and later to control wear on the blade after it has been in production for some time.

The quality assurance process consists mainly of three steps, where each step contains one or several CloudFlow services. The main challenge is to properly align the 3D scan data to the CAD model, which usually is a tedious manual process. A fully automated alignment is not necessarily feasible, especially if the CAD model has symmetries or the 3D scan is partial. The first step is therefore a coarse manual alignment, which is performed before an automated alignment process, where an optimization process iterates to make the point cloud fit as close as possible to the surfaces of the CAD model. Thereafter, the result of the registration needs to be reported to the user in an informative and user friendly manner. The entire workflow and the four software packages within it, is shown in Fig. 5.

1) Quality assurance pre-processing: The workflow starts of by letting the user choose the CAD model, 3D point cloud and where to store the results from the registration. This functionality is covered by the File Browser, mentioned in Section III-C.

Since CAD models can be stored in different file formats, and since the services later in the workflow are designed expecting a pre-defined file format, a file conversion might be needed at this point. If so, a conversion service accesses a dedicated virtual machine in the Cloud which runs the conversion. The conversion service is not guaranteed to finish within a HTTP time out, and is therefore implemented as an asynchronous service. The service launches the conversion as a background process, and WFM polls on the service to check if the process is finished or not. Before the conversion finishes, it provides a progress bar along with a text describing the current status. Note that this conversion requires no interaction from the user, allowing the WFM to automatically proceed to the next step of the workflow after the conversion is completed.

The manual alignment step, where the initial guess to the registration is made, is a web application. Here, the CAD model and the point cloud are shown in separate canvases and the user is expected to match corresponding points from the two models. Since the models can be quite large, a hybrid rendering is done through the Tinia framework [11]. The models are rendered server side, generating 3D images that are sent to the web client. The user can freely interact with the local model for an interactive experience, without transferring the CAD model. Similarly to the file browser, WFM awaits a message from the client to proceed in the workflow, and this signal is sent when the user accepts the initial guess.

2) Registration: The registration application is implemented to take advantage of parallel execution, and in order to integrate the application in the CloudFlow Infrastructure, the HPC service along with its design pattern described in Section III-E is used. A registration pre-processing service is implemented to generate the set of command lines required to run the registration based on the file IDs obtained in the file chooser applications, converter and initial guess application. A post-processing service for the registration is also implemented in order to enable semantic descriptions to the result from the HPC service. In this case, it will be the file ID holding the registration results. These three services are then stored as an

HPC registration sub-workflow, before it is modelled into the rest of the quality assurance workflow.

The registration has also been implemented as a single asynchronous service where the registration is run in the cloud environment instead of on the HPC cluster. Since the registration HPC sub-workflow has application specific inputs and outputs, the sub-workflow can be exchanged with the single cloud service directly. This can be useful as a cheaper alternative for users who do not prioritize performance. The service or the HPC block could potentially also be chosen dynamically. If the appropriate HPC queue is filled the execution in the virtual environment can be faster as it does not have to wait in the queue, even though execution itself is slower.

3) Quality assurance post-processing: The results of the registration is visualized by a WebGL application showing both the CAD model and the aligned point cloud in the same view. The distance between the models are illustrated both trough statistical information and color coding of the point cloud. The user will typically view and inspect these results for an unspecified time. The viewer is therefore implemented as a web application, where the user presses a "finish" button to tell WFM that the workflow should move to the next step. As there are no more next steps, the workflow is completed with a list of workflow output parameters. This list can be accessed later through the user's list of finished workflows.

V. Conclusion and Future Work

In this paper, the concept and the realization of a cloud-based platform is explained. The platform allows seamless integration and combination of engineering services, controlled execution, and monitoring of the used resources. The infrastructure components which build the platform utilize standards such as WSDL and SOAP in order to make the integration process simple. Additionally, employment of semantic descriptions allows discovery of compatible services and assists to chain services to form workflows. This type of assistance, semi-automatic orchestration, makes the services aware of each other improving interoperability.

The proposed solution has been validated by a wide range of end users solving different tasks from the manufacturing industries. Software providers will be able to offer their services in a common web portal. Software providers as well as consultancy companies can then combine available software solutions into potential workflows tailored to solve tasks provided by end users. The access to such efficient software solutions together with remote computational resources can dramatically reduce labor intensive tasks while improving the final product.

In the future, these semantic descriptions can be enhanced by expanding the semantic vocabulary so that WFE can orchestrate the creation of workflows fully automated, meaning without a user interaction provided that intermediary services exist. By intelligently orchestration any user can create their own tailored workflows consisting of their favorite software tools. So far, the solution is validated within the manufacturing industries. The technological choices are to little extent based on this choice, but validation in other segments will be needed to fully validate the solution.

Acknowledgement

This research was conducted in the context of the CloudFlow project, which is co-funded by the 7th Framework Program of the European Union, project number 609100. More information and news about CloudFlow can be found on the project website at `http://eu-cloudflow.eu/`.

References

[1] "UberCloud," https://www.theubercloud.com/, retrieved: August 2016.

[2] A. Bergmayr et al., "The evolution of CloudML and its manifestations," in Proceedings of the 3rd International Workshop on Model-Driven Engineering on and for the Cloud (CloudMDE), pp. 1–6, 2015.

[3] C. Stahl, E. Bellos, C. Altenhofen, and J. Hjelmervik, "Flexible integration of cloud-based engineering services using semantic technologies," in Industrial Technology (ICIT), 2015 IEEE International Conference on, pp. 1520–1525, 2015.

[4] "SimScale," https://www.simscale.com/, retrieved: August 2016.

[5] "cloudSME, simulation for manufacturing & engineering," http://cloudsme.eu/, Seventh Framework Programme (FP7) under grant agreement number 608886, retrieved: August 2016.

[6] J. Mendling, "Business Process Execution Language for Web Service (BPEL)," [Online]. Available: https://www.uni-ulm.de/fileadmin/website_uni_ulm/iui.emisa/Downloads/emisaforum06.pdf, retrieved: August 2016.

[7] M. Aslam, S. Auer, and J. Shen, "From BPEL4WS process model to full OWL-S ontology." [Online]. Available: http://bis.informatik.uni-leipzig.de/files/bpel_2_owls_short_paper.pdf, retrieved: August 2016.

[8] Oracle, "Oracle BPEL process manager datasheet,". [Online]. Available: http://www.oracle.com/technetwork/middleware/bpel/overview/ds-bpel-11gr1-1-134826.pdf, retrieved: August 2016

[9] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard, "Web Services Architecture," W3C Working Group Note, 2004. [Online]. Available: https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/, retrieved: August 2016.

[10] "Openstack Keystone," http://docs.openstack.org/developer/keystone/, retrieved: August 2016.

[11] C. Dyken et al., "A framework for OpenGL client-server rendering," in 4th IEEE International Conference on Cloud Computing Technology and Science Proceedings, CloudCom 2012, Taipei, Taiwan, pp. 729–734, 2012.