Customizing Scientific Data Analytic Platforms via SaaS Approach

Wenjun Wu  
State Key Laboratory of Software Development Environment, Beihang University  
Beijing, China  
wwj@nlsde.buaa.edu.cn

Chao Jin  
State Key Laboratory of Software Development Environment, Beihang University,  
Beijing, China  
jinchao@nlsde.buaa.edu.cn

Abstract—At the era of data driven science discovery, it is essential to provide customizable scientific data analytic platforms for researchers to conduct their personalized data intensive analysis. Science Gateway has been a viable solution to enabling scientists to run scientific simulations, data analysis, and visualization through their web browsers. But most science gateway frameworks are designed for integrating commonly used software tools and datasets in a specific science domain, thus requiring significant effort to implement the essential variability in lab-specific data processing workflows. In this paper we introduce a multitenancy architecture (MTA) based customization framework that can greatly accelerate the customization cycle of science gateway systems. Each tenant has his own workspace that assembles the software stack and tools to meet the software requirements of his specific data analytics tasks. Through this framework, developers can import their domain-specific analysis pipeline scripts and mashup relevant templates including GUI templates, tool recipes and workspace templates to generate both workspace and web interface for running these application workflows and visualizing the output from workflow executions without writing extra wrapping code.

Keywords—SaaS, Science Gateway, Workflow, Cloud,

I. INTRODUCTION

A science gateway is a scientific data analytics platform that includes a community-developed set of tools, applications, and data customized to meet the needs of a targeted community. Initiated in 2004, many scientific gateways funded by the TeraGrid Science Gateways program [1] have been developed to offer Software-as-a-Service (SaaS) to researchers from science domains such as bioinformatics, climate, and high energy physics. Large-scale scientific data analytics often require expert skills in both computational science and domain fields. Previously, scientists used to rely on their in-house IT staff to manage their small computational clusters, deploy computation software packages and perform data process tasks for their lab tasks. But in most cases, research labs lack of enough staff resource with all these skills to create environments for automated data analysis. Based on the SaaS delivery model, science gateways provide a viable solution to this problem by enabling scientists to run scientific simulations, data analysis and visualization on national HPC infrastructure through their web browsers without software installation and deployment.

Despite of the success of science gateways for general use, few of them have wide adoption for full-fledge data analysis pipelines in research labs. The problem is caused by the diverse nature of research lab settings. Scientists are more interested in customizing their research process in terms of scientific data analytics to achieve new insight into the same research subjects. Moreover, scientists keep modifying and refining their current data analysis methods to obtain better results during the research lifecycle. Most science gateways can only capture the commonality in tools and datasets among a research community, but fail to support the essential variability in the lab-specific data processing workflows. Therefore, the major design challenge in the development of a scientific data analytics platform is about how to facilitate rapid customization of computational software tools and data analysis methods originated from community towards the needs of individual researchers.

To tackle the customization issue, a few web-service based frameworks [2] [3] have been proposed to support deploying scientific applications on computational resources and wrapping these applications as web services for workflow composition. These software frameworks can streamline the process of application integration and simplify customization effort. Recent science gateways such as Galaxy [4], demonstrates more customization capacity by introducing both application configuration framework and community repository to incorporate third-party scientific software tools. But it often takes a lot of development effort to achieve customization in Galaxy framework because it supports homogenous runtime environments with the same software stack and distributed computing platform. When scientists want to add a new application tool in their science gateways, developers still have to build wrappers and even refactor the original implementation of the tool in the Galaxy development environment.

This paper proposes a new SaaS based customization framework that allows developers and researchers to quickly deploy and personalize their data analysis pipelines on cloud resources. Through this customization framework, science gateway developers can focus on defining computational workflows for domain-specific applications and use the software tools in the framework to automatically generate SaaS interfaces to running the workflows and visualizing the outputs.
from workflow executions. In the integration and deployment process, developers merely specify the run-time workspace for each application without modifying the implementation of these applications.

The paper is organized as follows: Section 2 elaborates the customization problem of science gateway platform by giving an in-depth discussion about science gateways in the domain of next generation sequencing. Section 3 proposes the SaaS oriented customization framework for science gateways. Section 4 introduces the workspace and workflow management in this framework. Section 5 summarizes our research work and point out future research topics.

II. PROBLEM STATEMENT

The design ration of science gateways is to capture the common requirements in a research community and integrate software tools and data that enable scientists to perform their data analysis tasks. In a relatively mature science disciplines such as bioinformatics, there are many off-the-shelf software tools available, which are either free or commercial, for scientists to choose and chain them together to construct analytic pipelines. This is often a trial-and-test process in which scientists start with the most general pipeline that are well adopted by a research community, and then try to refine and adjust the pipeline toward their lab’s needs. It demands customization machinery to be built in a science gateway in order to smoothly facilitate researchers to customize the original pipelines and evaluate the derivations. However, most state-of-art science gateways only have limited capacity to support flexible customization. In this section, we elaborate this situation in the context of next generation sequencing (NGS) in life science, and discuss customization features in current science gateways.

A. Customization of Data Analytics in NGS

With revolutionary sequencing technology, NGS provides a much less expensive way to perform personal sequencing, which promotes the democratization of genomic sequencing around the world. Currently there are approximately 1500 NGS platforms of all types installed world-wide, a number that is projected to triple in the next 12-18 months. The explosive growth in NGS devices brings huge amount of sequencing data for scientists. For example, the HiSeq 2000 from Illumina can produce 200 Gigabase-pairs (Gbp) of sequence data over the course of an eight or nine day run, during which it generates 40 TB of raw instrument image data that must undergo image and signal processing before human-intelligible polynucleotide sequencing “read” data is produced. Very large-scale NGS data production and analyses have transformed these disciplines into data-intensive science.

Figure 1 illustrates the most common NGS analysis pipeline—the mapping of NGS short reads onto a “reference” DNA sequence dataset (e.g., a human genome reference sequence). It consists of three major steps including vendor-specific base calling, alignment and assembly, analysis and visualization.

(1) Image analysis and base calling: The NGS analysis pipeline starts with image analysis and base-calling, which classifies the instrument image output signal intensities into DNA base calls for each nucleotide position in each sequencing read. The base-calling step is performed by software package provided by sequencer vendors such as Illumina, 454 life sciences and SOLiD. The analysis task can be accomplished either offline or on-the-fly on the workstation or small compute cluster that is typically part of an NGS instrument. The output from the base-calling step is DNA sequence reads, which is often kept in a standardized FASTQ file.

(2) Alignment and Assembly: The entire set of these short reads is mapped (or assembled) by DNA sequence alignment onto a reference DNA sequence by a DNA sequence assembler to yield a set of contigs, which although they may be separated by DNA sequence gaps may be ordered and oriented into scaffolds. There are two types of DNA sequence assemblers: a mapping (or comparative) assembler and a de novo assembler. A mapping assembler aligns the reads onto a reference sequence, whereas a de novo assembler does not use a reference sequence to inform the overlap and assembly of reads. Mapping reads to a large genome can be very computationally intensive—to assemble a human genome may require 200,000 CPU-hours to finish the read mapping process with very short reads. When researchers sequence the genome of a new species without a reference DNA sequence, they have to use de novo assembly, which with short reads is typically an NP-hard graph optimization problem.

(3) Analysis and Visualization: The assembled DNA sequence data need to be further analyzed to identify their structure and functions. The most common NGS analysis is identification of SNP (Single Nucleotide – polymorphism) to reveal genetic variation in DNA sequence. This step is normally named as SNP calling that compares the reference genome with the consensus sequence and refine all variant calls to improve accuracy. Since the preliminary SNP calling often produces ambiguous SNP results, the subsequent SNP filter masks these data to create an improved SNP call set. This set is used as training data for variant score recalibration that aims at improving the biological variant estimation. The generated results are reported as Variant Call Format (VCF), which can be further visualized in Genome Browser tracks with the DNA sequence and other annotations.

Towards the NGS data analysis task, there are a wide variety of freely available tools for NGS short-read alignment and variant calling. The NGS community web site—SeqAnswers [5] lists more than 600 NGS software packages that are commonly used by NGS community. Essentially these
tools demonstrate two aspects of variability: function and software environment.

(1) Functional Variability: Typically, if a NGS algorithm and tool is developed by researchers working at famous life science labs, it may have good chance to become a popular and widely accepted tool. But these software tools may not always cater to the needs of an individual biologist who operates at a much lower capacity, scale and magnitude. That’s why life scientists like to evaluate common solutions, identify the best ones, and then introduce variants into them, including new analysis algorithms and different databases for lab work.

For instance, the exome sequencing was introduced as a more cost-efficient alternative approach to the whole genome sequencing [6]. Multiple tools including alignment and SNP calling [7][8] for exome sequencing studies have been implemented in the community. Research efforts on a combinatorial approach [9] have been made to verify the reasonable combination on exome sequencing analysis. The experimental result suggests that the four combinations of tools at alignment and post alignment stage can demonstrate high accuracy and sensitivity to reduce the number of false positive variants. In addition to the variability in the steps of alignment and SNP calling, researchers often write in-house visualization scripts to extract the corresponding average base qualities of the filtered SNPs and plot the result in a customized way.

(2) Variability in Software Environments

Each NGS tools may depend upon different software libraries and even operating systems. According to the tool list of SeqAnswers [10], NGS tools are implemented in nearly every mainstream programming languages such as C, C++, Java, Python and Perl. Majority of the tools are able to run on Linux but a few tools can only run on Windows Platform. Furthermore, because some NGS tools are very computational intensive applications, they are implemented in parallel computing environments such as multicore machines, MPI clusters and Hadoop platform. All these variants in the aspect of software environment demand different settings of software stack and underlying infrastructure.

B. Limitations in Current Customization Frameworks

Galaxy [4] is a widely used science gateway for performing accessible, reproducible, and transparent genomic science. It offers life scientists with composable bioinformatic functionality, especially NGS-related functionality via Galaxy tool integration and workflow framework. Through Galaxy, users can access a variety of NGS analysis tools, construct and run their data processing pipelines. Galaxy offers the follow features to support customization:

(1) Plug-In framework for tool integration

Galaxy introduces a plug-in framework to integrate NGS tools contributed from the research community. When a Galaxy developer tries to build a Galaxy application plugin based on a third-party tool, he needs to firstly rebuild the tool and test it under the development environment of Galaxy. After the tool is validated, the developer can define a XML configuration file for the tool and upload the file onto the Galaxy instance in his lab. By parsing the XML file, the Galaxy server can generate a simple web interface and lay out the interface in the main web portal of the server. The XML file also tells the server how to invoke the tool with the right command-line options and what data files it will produce as output results.

(2) Galaxy tool shed for sharing verified tools in the NGS community

In addition to the above “Plug-In” feature, Galaxy introduces a mercurial repository named by Galaxy tool shed to enable sharing functionally correct Galaxy tools (including workflows and data ) for the Galaxy community. Scientists can deploy Galaxy server instances in their own lab and customize their in-house Galaxy by selecting the appropriate Galaxy tools from the Galaxy tool shed. Via the interface of mercurial version control system, Galaxy tool shed supports sharing updates to the tool code among multiple developers throughout the process.

(3) Workflow editor and engine for composition of customized analysis pipeline

Galaxy also offers users a Web-based workflow editor to chain up multiple Galaxy tools into their data processing pipelines. Behind the scene, Galaxy’s workflow engine can invoke each tool in a Galaxy workflow and track every output data from the execution of the tools. This data governance feature allows a user to record every step of his workflow and its intermediate results and share them among his collaborators.

Although Galaxy tool integration framework supports plug-In style customization, it has the major problem in customizing service platforms in terms of both functional and computational variability. The design assumption of the Galaxy framework is to facilitate researchers to set up their own instances of the Galaxy server on their in-house computing infrastructure. Tight integration of the Galaxy workflow system with the toolshed makes it easy for an administrator to install a tool from a toolshed directly into a Galaxy instance. While a standalone Galaxy server can be beneficial to individual researchers, it needs sufficient amount local computing resource, heavy administrative effort and high IT expertise. Given the large scale of high-through sequence data generated by the latest models of sequencers, small bioinformatics labs can’t afford very expensive computing clusters to analyze the data. Even if their lab budget can cover the computing cost, administering a Galaxy instance and managing hundreds of complex genomic tools within Galaxy can be extremely cumbersome and requires computational expertise beyond most of the groups.

Ideally, researchers in the NGS community prefer to run their customized analysis workflows on a public Galaxy server to avoid administration overhead with toolshed and local galaxy management. Currently, 54 Galaxy server instances have been deployed around the world [11], where Internet users can access and utilize the tools included in the Galaxy toolshed. It also allows users to specify their Amazon EC2 credentials to launch tools in the Amazon Cloud infrastructure via CloudMan [12]. However, the tool management mechanism in those Galaxy servers only supports system-level customization. Without administrative privilege, individual users have no way to create private toolshed and upload their
own applications tools onto the repositories of public Galaxy servers. Hence, there is a need for multi-tenant tool management in Galaxy services that can offer private workspace for user to manage their tools and data.

In addition to the lack of support functional variability for users, Galaxy customization framework can’t address the issue of computational variability either. It is well known that bioinformatics workflows are typically composed of many diverse executables, each with dependencies for specific packages or software environment. For example, some NGS tools listed in SeqAnswers can only run on a Windows platform, which makes it impossible to be directly hosted on a Linux box required by a Galaxy server. Furthermore, Galaxy host parallel data process applications on cluster environments, which is often managed by distributed resource manager such as PBS and SGE. But all these DRMAA [13] based distributed computing framework are not completely compatible with Hadoop, thus unable to support Map-Reduced based NGS tools.

III. SaaS ORIENTED CUSTOMIZATION FRAMEWORK

To enhance customization capacity of domain-specific science gateway platforms, we introduce a novel MTA based framework named as **Easy ScienceSaaS**, by leveraging the latest research in both SaaS [14] and Devops [15][16]. It extends the grapevine model of EasySaaS framework [14] towards the domains of scientific data analytics by adopting multi-tenant workspace as a core notion to describe the variability of computational requirements. Each tenant has his own workspace to host his analysis pipelines and datasets to satisfy his specific requirements on functional and computational aspects.

As shown in Figure 2, a four-layer template structure is defined in the framework. Tenants are able to customize their web user interface, analysis pipeline, computational tools and workspaces. Conceptually, a tenant workspace consists of description of bioinformatics tools software stack and runtime environment, which can be transformed into virtual machine (VM) instances and software artifacts in the virtualized computational infrastructure. Both the software stack and runtime environment of a tenant’s workspace are determined by his computational workflows. Every workflow often involves multiple steps of data processing, each of which calls a software tool with specific software dependence. In addition to the application-level software stack, every tool also demands the appropriate runtime environments such as underlying operating system and distributed computing infrastructure. Especially when a tool is a parallel program, it may need data-intensive computing frameworks such as MPI or Hadoop. Thus, a tenant workspace’s description is actually an aggregation of all the software requirements of the tools in his pipelines.

![Figure 2 Easy ScienceSaaS Customization Framework for Science Gateway](image-url)

A. Workspace Template

A workspace template is designed to program and generate virtual computing infrastructures for tenants. It defines the virtual topology of a computing cluster and specifies the virtual machine’s hardware feature of each computing node as well as the mechanism of distributed resource management.

Infrastructure automation frameworks such as Chef [15] and Puppet [16], introduce domain-specific language (DSL) for writing system configuration "recipes" in a "cookbook" and streamline the task of configuring computational resources. Our Easy ScienceSaaS framework adopts Chef and its template language to describe workspace cookbooks. To construct a virtual cluster for the workspace, the Chef recipes often describe three steps of configuration: installation of cluster software packages on each compute node, configuration of the virtual network connecting these compute nodes and initialization of proper service daemons on both head node and worker nodes. Given the limited types of virtual clusters, community developers can separately define recipe templates of different clusters such as PBS/Torque cluster and Hadoop cluster. Figure 3 illustrates a recipe segment for specifying a Hadoop cluster.
name  "hadoop"
description  "Installs hadoop and sets up basic cluster per Cloudera's quick start docs"
depends  "java"
# Hadoop supports both debian and ubuntu
%w{ debian ubuntu } each do |os|
supports os
# Installs hadoop from Cloudera's repo
execute "apt-get update" do
action :nothing
end
template "/etc/apt/sources.list.d/cloudera.list" do
owner "root"
mode "0644",
source "cloudera.list.erb"
notifies :run, resources("execute[apt-get update]"), :immediately
end
execute "curl -s http://archive.cloudera.com/debian/archive.key | apt-key add -" do
not_if "apt-key export 'Cloudera Apt Repository'
end
# start all the services on Hadoop nodes
%w{namenode secondarynamenode datanode jobtracker
tasktracker}.each do |d|
service "hadoop-#{d}" do
action [:start, :enable]
end
end
end
Figure 3. A cookbook for a Hadoop cluster template

The above example only presents a basic software environment for running Hadoop based analysis programs. Users can further extend this template into a complete domain-specific Chef cookbook by integrating more recipes that describes their data processing tools towards their analysis pipelines.

B. Tool Template for command-line science applications

In the NGS domain, every tool is actually single or multiple command-line programs for DNA sequence analysis. Our previous research [3] already proposed a metadata scheme to describe the command syntax and invocation stub of a tool. The command-line description enumerates the parameters of the program, through which the Easy ScienceSaaS framework can generate a web widget of the tool and plug the widget into the whole web interface. And the invocation stub lists the execution script, input and output data types, which enables the workflow engine to run the tool. In order to link a template of an analysis tool with its workspace template, we append the tool’s meta-data description with its Chef recipe that specifies the runtime environment. Figure 5 demonstrates an example about Hadoop Blast, which requires a Hadoop environment and C++ libraries as well as the Blast database.

Figure 4. Blast Template enriched with its Chef recipe

C. Workflow and Web Interface Template for data analysis pipeline

A workflow template is an ensemble of the essential data process tools and their interconnections. Formally, it can be conceptualized as a Direct Acyclic Graph, where each node represents a data processing unit calling a designated external program. Each node is a three-element tuple with the attributes include: Data Process Unit, Input Ports and Output Ports. The input ports links with the output ports of upstream data process units, and the output ports links with the downstream ones.

Normally, for a generic data processing task, there are multiple common workflow templates. For instance, the exome analysis can adopt a workflow template with the GTK toolkit as its core processing unit. The workflow template also provides variant points for users to make their customizations, such as different calling parameters, customized analysis tools, GUI and visualization scheme.

Customization of workflow GUIs is implemented via web interface templates that encapsulate the common functions including web forms for setting parameters, visual elements for displaying status of workflow execution and intermediate results. Tenants can utilize these templates to generate the actual interface to the workflows by integrating the web widget of each data analysis step in the workflow.

IV. WORKSPACE AND WORKFLOW MANAGEMENT

In our Easy ScienceSaaS customization framework, a tenant’s workspace cookbook is generated by merging all the recipes of the tools in the tenant’s workflow. But given the diverse sources of these tool recipes, there may be inconsistency, or even conflicts among these recipes. Thus, without extensive combinatorial validation of these recipes, it is very problematic to automatically consolidate these recipes into a single cookbook. In this section, we introduce a detection and mediation process in Workspace Management to avoid possible conflicts of the recipes. Based on this method, the system can establish a tenant workspace that contains heterogeneous virtual clusters with multiple types of virtual machines. Because each VM encapsulates a software environment for individual tools in a pipeline, the workflow engine of the system need to effectively coordinate the execution of the pipeline across different types of VMs. We design a new workflow engine named as GreenPipe [17] that can manage the computational processes running on separate VM instances and perform on-demand resource provisions.

A. Configuration and Deployment of Tenant Workspace

Figure 5 illustrates the major components and basic steps in the process of customization and deployment in the Easy ScienceSaaS framework. The central component of the framework is the community repository, a database preserving metadata and packages of web interface, workflow, tool and workspace. Every user can publish his tool templates and packages to share them in the community. The runtime environment of our framework consists of three major components: Web container, GreenPipe workflow engine and enhanced Chef based configuration service. They are responsible for transforming metadata description and
templates into real objects for representational rendering, workflow execution, and VM contextualization. Pulling the cookbook template of a workspace from the repository, Chef service can instantiate pre-defined VM templates and set up software environments on the VMs.

For a set of recipe, \( RS = \{ RS_1, ..., RS_p \} \). \( RS \) is a feasible set, if and only if each \( RS_i \), \( RS_j \in RS \) is compatible. Apparently, recipes in a feasible set can be safely merged. Thus, a workspace description is actually a group of such recipe sets: \( \{ FS_1, FS_2, ..., FS_w \} \), where each \( FS_i \) is a feasible set.

We design a conflict detection method to scan all the recipe pairs among the \( N \) tool recipes and find the potentially problematic pairs that can’t be combined in a Chef cookbook. By filtering all these recipe pairs, we can partition the \( N \) recipes into multiple feasible sets. Each set is represented as a Chef cookbook that is used to generate a group of VM nodes with the same type of the VM image.

(4) Workspace Deployment

Based on the workflow template, the system can automatically configure its web interface in the form of web widgets. The web container is responsible for laying out these web widgets in the tenant’s web browser. Based on the workspace cookbooks, the Chef Service can perform configuration and deployment of software stacks on the pre-defined VM images. The initiated VM instances are organized as a heterogeneous virtual cluster for the workflow engine to launch the real analysis pipeline. After the VM clusters are in the state of ready, the workflow engine starts to compile the abstract workflow and produce an execution plan on the clusters.

B. Workflow Management

The GreenPipe workflow engine coordinates the invocations and data transfers among the VMs of the heterogeneous virtual cluster. During runtime, it launches VM instances via Chef Service and then assigns the execution of analysis tools onto the appropriate VM instances. All the output data from the execution of a workflow are kept in the persistent storage to track the provenance of the workflow. Figure 6 displays the control structure of the GreenPipe workflow engine including three important components: VM Provisioner, Job Control and Workflow Monitor. VM Provisioner interacts with the Chef Service to manage VM instances. Job Controller polls workflow jobs from the job queue, and runs execution workers to perform data staging and processing on VM instances. Workflow Monitor checks the status of workflow execution and keeps UI posted with up-to-date status reports.
When a workflow execution request is pending, VM Provisioner checks the workspace cookbook of the workflow and pass the cookbook down to Chef Service to initialize a virtual cluster for running the workflow. If this is the first time for Chef Service to transform the cookbook into a virtual cluster, Chef Service will generate VM instances from basic VM templates and perform all the package downloading and configurations on those instances. Otherwise, Chef Service can locate cached VM templates and only perform necessary configurations on the templates. During the active period of the instances, the controller constantly updates the runtime status of the instances in the database, which is used for monitoring and reporting purpose.

When a virtual cluster is completely initialized, Job Controller will kick off workflow execution and start to launch workflow execution workers for running actual data processing on the virtual cluster. For each step of the workflow, Job Controller finds the appropriate VM nodes in the cluster to host the workflow workers, uploads the input data from the persistent storage to the VM nodes, and copies the output result from the workers back to the storage for the next step of workflow execution. When the workflow is completed, the VM Provisioner requests Chef Service to release all the relevant resources utilized by the workflows and shutdown the VMs.

V. CONCLUSION

Science Gateway has become an essential scientific data analytics platform for domain scientists to make significant discovery from massive scientific data. But in most science gateway platforms, it is still very cumbersome for individual researchers to set up customized computational environment with their own data analysis methods and software tools and originated from community towards the needs of individual researchers. We propose a new customization framework named as Easy ScienceSaaS to alleviate the workload of science gateway developers and provide an easy approach for customization according to tenants’ requirements in a collaborative manner. In the four-phase customization process in the framework, a tenant can begin with a group of templates from the community repository, mashup the templates with his personalized tools to compose his data analytic pipelines, and establish hosting workspace for the pipelines. We also discuss the utility of the framework with an example from the domain of next generation sequencing.

ACKNOWLEDGMENT

This project is sponsored by National High-Tech R&D Program of China (Grant No. 2013AA01A210) and the State Key Laboratory of Software Development Environment (Funding No. SKLSDE-2013ZX-03).

REFERENCES

[3] Wenjun Wu, Tom Uram, Michael E. Papka, Accelerating science gateways development with Web 2.0 and Swift, TeraGrid 2010