Enabling User Driven Big Data Application on Remote Computing Resources

Weijia Xu
Texas Advanced Computing Center
The University of Texas at Austin
Austin, Texas, USA
xwj@tacc.utexas.edu

Ruizhu Huang
Texas Advanced Computing Center
The University of Texas at Austin
Austin, Texas, USA
rhuang@tacc.utexas.edu

Yige Wang
Department of Computer Science
The University of Texas at Austin
Austin, Texas, USA
yige.wang@utexas.edu

Abstract— **Driven by the computing resource requirement, there are increasing demands of migrating data driven analysis from local computing resource to powerful remote resources such as cloud and high performance computing cluster. In addition to various commercial cloud services, there are also rich selections of high performance computing centers in academia providing cyberinfrastructure (CI) offerings. However, access barriers exist in bring those resources to data driven research community at large. To help lower those access barriers and increase the adoption of utilization of remote resources for data driven analysis, we propose a new service model for utilizing remote computing resources, which empower users to deploy and run their big data application as a web application on remote computing resources. There are several key design goals of this model including *enabling interactivity, reusability and reproducibility*. Compare to the traditional batch-processing model commonly supported by CI resource providers, supporting a web application interface enables interactive analysis capabilities. Users design the application through a configuration file utilizing a set of pre-defined task templates that are also extensible by users. The application generated from the configuration file is self-contained and can be deployed without alleviated system privilege. Therefore, ad-hoc analysis routines can be described and preserved in a format that can be shared and re-used. Remote resources can also be described and implemented through configuration files to automatically bridge the application with remote resources and facilitate migration with different resources in the future. Consequently, analysis tasks can be preserved through the configuration file for reproducibility. Here we detail our proposed application framework and its preliminary implementations. We demonstrated usage of this framework with a practical use case of aggregating and analyzing live tweets.**

Keywords-component: high performance computing; cyberinfrastructure; reconfigurable web service; service computing; big data

I. INTRODUCTION

One of profound transformations brought by big data in recent years is the increasing adoption of data driven analytic methods across different domain fields. Fueled by massive amounts of complex data produced by businesses, scientific applications, government agencies and social applications, data-driven analytics have the potential to help users gain new insights for decision making, scientific discovery, business insight, marketing potentials and more [1]. However, a facet in realizing this promise has been the new techniques and tools for conquering extremely large datasets with the latest advances of computing infrastructures [2] [3] [4]. Despite rapid advances in hardware and software technologies have significantly speed up the analysis and lowered the analytic barriers, big data analysis in practice commonly requires significant computing resources throughout its lifecycle, from data aggregation, management to analysis.

With the emerging of data driven analysis and data science, the usage of high-performance computing resource and cyberinfrastructure (CI) has spread across almost all fields in academia. There is an increasing trend to move computational expensive analytics from standalone workstations to remote advanced computing resources such as a commercial cloud or cyberinfrastructure funded by various agencies and maintained in various high performance computing centers nationwide [5] [6] [7]. Those centralized remote resources provide cost effective solutions for researchers and facilitate new discovery.

While CI providers have continued success with infrastructure-as-a-service (IaaS) model, there are increasing demands to offer more service models for diverse use cases. Along with the demands, new service models, such as data science as service (DSAS) and machine learning as service (MLAS), are proposed. Yet, unique characteristics of big data analysis, such as data centric computing, explorative nature and interactivity, present challenges with existing remote resource access, request and allocation models commonly used in practice. The complexity of infrastructure and current access models present challenges to big data analysis needs especially for those who are not used to remote computing environments [8]. Hence, a new service model of remote computing resource is needed to serve data driven analysis with more flexibility and better usability.

Here, we explore a new model for utilizing remote resources to improve the CI accessibility for wide variety of use cases and domain fields. The proposed model has the promise to dynamically meet various analysis needs by enabling users to set up their own interactive web application interface, which interoperates with existing resource and services. The proposed framework follows model-view-controller design pattern. Common operations are abstracted as *task*s. In addition to data model, each *task* also has a *view* template for web application interface. Multiple tasks can be composed to an *application* through a simple configuration file in JSON format to describe operations required to accomplish a use case. The framework can then generate a web user interface based on configuration file to allow user

running the analysis interactively on remote resources. The framework also includes pre-built support of provisioning resources, dynamic credential and access control and self-contained web server that can run without requiring alleviated system privileges. We demonstrate the proposed framework through a use case demonstration of aggregating and analyzing live tweets using multiple programming environment and tools.

The approach presented here is different from existing support tools and models of using cloud and high-performance computing resources with following contribution highlights:

- Enabling users to compose and personalized web application on demand.
- Giving users full control on the web application execution and support interactive usages.
- On-demand user credential creation and management
- Interfacing with existing analysis tools
- Scalable to additional use cases and resources.

The rest of the paper is organized as following. In Section II, we review existing usage models of existing cyberinfrastructure and how our proposed approach differs from those existing approaches in details. Section III describes the design and implementation of proposed framework design and its key components. A use case of supporting Twitter analysis using the proposed framework is detailed in Section IV. We conclude and briefly discuss ongoing works in Section V.

II. Backgound and Related Work

A. Cyberinfrastructure Resources Access Models

In academia, CI has been increasingly used in open science research and enabled break-through discovery in many domains. CI providers now face challenges to meet growing demands. We identify three utilization models that are most common within CI communities for open science.

Job Submission with Secure Shell Connection This mode requires users connecting to the CI, typically through a dedicated login node on remote source, using a client application via Secure Shell Connection (SSH) protocol. After a user logs onto the CI, the user can start a ***job*** through an editable script, which describes resources requested and computations to be finished. There are several software tools for managing job and resource allocation such as SLURM, OGS etc. [9]. After the job submissions, users can use the commands to check the job status as determined by the resource/job manager, such as in queue, running, finished etc. All interaction between user, CI and their application is through the command line interface (CLI).

The access model follows infrastructure-as-a-service (IaaS) mode. The physical and/or virtual resources are allocated to users upon request. However this access model has limited interactive analysis support and does not provide graphical user interface. The CLI limits usability of the CI resources and distances itself from users in non-computational fields. Furthermore, analysis can often be facilitated with comprehensive user environment and visualization support, which can be supported in other models.

Remote Software Session In this scenario, a user can connect to software service running at remote CI resources using client software installed locally. The user may have to manually start the software service at remote resource before making the connection. A common example of this model is through Virtual Network Computing (VNC) session [10]. In this session, the user run VNC client locally in order to connect to the VNC server session running on the remote resource. The VNC session allows applications with graphical user interface or visualization to be used remotely. Other examples include using SQL client to connect to remote database for data operations, using web browser to access Rstudio session for interactive analysis [11].

This approach is effectively a software-as-a-service model (SaaS). Although analysis environment and remote visualization can be supported, both availability and accessibility are still limited. Generic support software tools, such as VNC, also suffer performance issue due to limited network bandwidth. The interoperability with additional resources and software tools and share among users are either not possible or very hard to implement by users.

Community Portal and Gateway A community portal (also known as, gateway, user portal) is a website that dedicated to users from a particular domain or sharing common needs on specific resources. The website has credential management that allow users to log on. A web portal often contains a set of features and tools that can simplify the process of using CI for a set of common use cases for its users. One of the advantages of the portal access is the greatly improved usability as it can outfit an otherwise command line interface into a web based user interface. A successful example of community portal is the CyVerse project [5].

In this model, a web portal provides a platform through which remote resources are available to users. Therefore, it is a form of platform-as-a-service. Through this platform, applications can be adopted to start within the portal and increase the accessibility and usability of CI resources for end users. But community portal incurs large up front costs for initial software infrastructure development and deployment. Once in production, significant efforts and resources are required to maintain its operations, expand its features and support evolving use cases. Community portals also form a potential point of failure due to maintenance downtime which can temporarily block users from using the resources. To adopt an existing implementation for a new use case deployment remains a challenging process and requires in-depth technical expertise.

The proposed application framework is compatible with three service models described above while offers complementary features. With IaaS model, users can start a customized web application easily using allocated resources with improved usability and interactivity. For CI providers, the framework provides a viable solution with low development overhead. For example, CI providers can deploy an implementation of proposed framework for applications for specific groups of users. The framework can also be deployed and made available as part of platform to support dynamically created web applications.

B. *Scientific Workflow Management*

The framework related to widely available existing workflow management system from open source community and industry [12] [13] [14] [15]. The framework presented here shares similarity with existing scientific workflow tools in customizable and reconfigurable by users. However, a key difference is that the proposed framework is designed to run as a standalone web application. Therefore, it doesn't rely on server-side software deployment at remote resources. Hence the proposed framework can be easily run on different remote resources and even with the potential to run in a heterogonous resources environment. Additionally, a key component of the proposed framework is the credential management that enables fine-grained access control and the ability to generate new credential dynamically. This component enables not only the defined *workflow* which can be shared as a file but also the analytic instance can be accessed by multiple users.

III. ARCHITECHURE DESIGN AND DATA MODEL

A. *Architecture Overview*

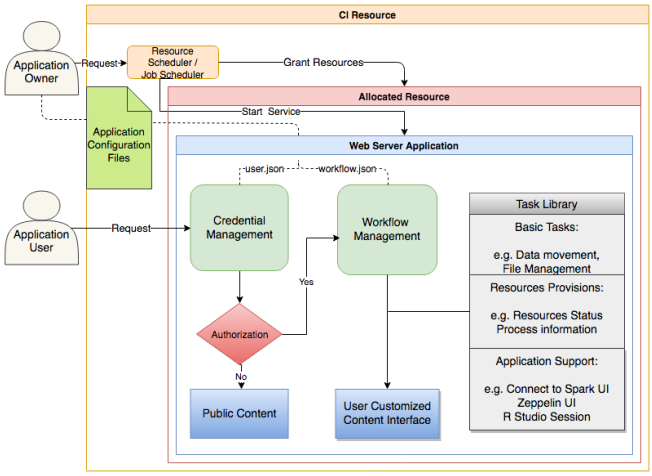

Figure 1. Overview of framework architecture and access workflow

An overview of the system architecture of proposed framework and startup workflow is shown in Figure 1. In Figure 1, *application owner* refers to person to start the web application service. A*pplication user* refers to individual who will access and utilize the web application service. To start the web application, application owner will first request resources from remote cyberinfrastructure. Once the application and its configuration files are accessible from the allocated resource, a web application server is started on the allocated resource for application users to interact through web interface.

The web application can be started by resource provider and available as a service directly to application users (SaaS). The proposed framework also enables application user to start the web application. In this case, application user is also the owner of the application. And the resources providers just provide their infrastructure as a service for the user.

There are four key components of the proposed web application framework: credential management, application management, task libraries and application configuration files. The application configuration files are central in the proposed web framework. There are several types of configuration files for application parameters, user configuration and workflow configuration. Each of the configuration files is in JSON format with a list of predefined fields and values. Many features and content of the web application can be dynamically and customized through configuration files. For example, the credential management component can use configuration file to initialize application user credentials. The workflow management component can generate customized workflow composed by tasks, which are pre-defined in task libraries and specified through configuration file.

B. *Credential Management.*

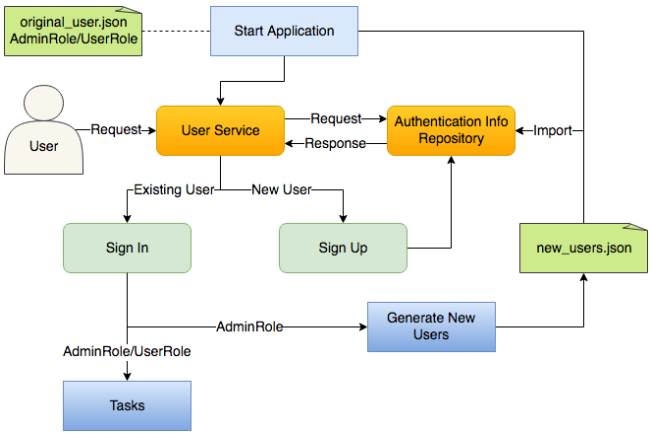

Figure 2. Overview of credential management component

The credential management provides access control support to other services available through the web application. There are two key modules, the user service module and the authentication module. The user service module supports common user management functions such as sign in, sign up and create new user accounts. The authentication module includes authentication service interface and secure credential repository management. Figure-2 shows an access workflow of credential management component.

Each user object includes several basic fields including *name*, *email*, *password* and *role*. Additional authentication provider specific fields are also available and extensible for future use cases. Users can be defined using configuration files in JSON format and available upon start of the web application. Figure-3 shows an example of a basic user object and corresponding JSON data.

There are two types roles currently implemented: *AdminRole* and *UserRole*. *AdminRole* users can have access to the set of tasks that are not available to regular users. One of such role-controlled tasks is to generate additional temporary regular user accounts on-demand. This feature is especially designed for a situation where application owner wants to share application with other users on the fly. As

described in the next section, availability of tasks can also be configured and limited based on user roles.

```
{
  "users" :
    [
      {
        "firstName":"Admin",
        "lastName":"User",
        "password":"1111",
        "email":"admin@utexas.edu",
        "role":"AdminRole"
      },
      {
        "firstName":"General",
        "lastName":"User",
        "password":"1111",
        "email":"general@utexas.edu",
        "role":"UserRole"
      }
    ]
}
```

Figure 3. User definition and exemplar user configuration files.

In addition to manage credential locally on the server side, the authentication module can be configured to use remote authentication service provider using OAuth2 protocol. Each user object can be linked with additional service provider accounts or social credentials. Similar to other OAuth 2.0 supported providers; an access token is provided to the framework for each credential service. With the access token, user information such as name and email are retrieved. The framework saves all dynamic generated users to a file with limited access permission. To guarantee no user account is duplicated, the framework uses user emails as a unique identifier and checks both service module and file for user emails.

It is important to note that the main purpose of user credential management is to enable flexible access of the application while it is running in order to share its process for online collaboration. Therefore, external credential is only used during the application and not saved for future access.

C. Task Libraries

The task libraries include a number of pre-built action modules that can be customized and re-used to compose different workflows. Each task module includes a data model, which holds information and actions of this task module, and an HTML section template, which serves as basis for dynamically rendering web content. Application management component and tasks together are the key elements of conventional Model-View-Controller design pattern [16].

Figure-4 shows a hierarchical view of selected task modules in the proposed framework. Based on their functions, tasks are grouped into three categories: basic tasks, resources provision tasks, and application support tasks. The basic task group includes tasks to support common file manipulations and display static content from server side to the client. The resources provision tasks refer to those tasks help users to interact with the remote computing resources such as connecting, checking cluster status and previously submitted job status. The analysis tasks are the set of tasks that will running an external application, such as a bash script, MPI job, and launch Zeppelin notebook.

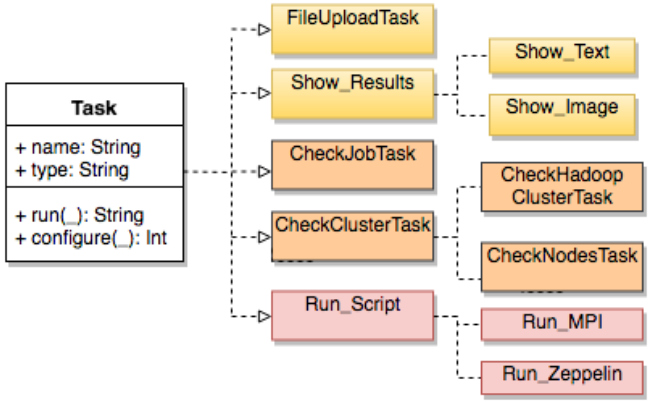

Figure 4. Selected task modules in the proposed application framework

Each task contains several common fields and methods extended from a base class. The *configure* function parses an input JSON object to set appearance and values of each task. The *run* function will start a set of actions on the server side when an execution request of this task is received. Each implemented task is mapped to a specific action and may require additional parameters or methods. Therefore, the web interface template for each task can vary from each other. A task module and its field values can be specified through a configuration file. Multiple task modules specified in the configuration file will be used to compose a customized web application interface.

D. Application Management Component

The workflow management component is responsible to create a dynamic web application interface based on a given workflow configuration file in JSON format.

An exemplar application configuration file consisting of three tasks is shown in Figure-4 (left). The result of this configuration file is a web interface with three steps each of which corresponds to one task (Figure-4 right). This exemplar workflow is designed to help users to run data parallel tasks. The application requires users to provide two script files, split.csh and combine.csh. The split.csh will automatically split the data into multiple parts and start multiple processes to run in parallel. The combine.csh will synthesize final results from distributed executions. The example shown here is a simplified extraction of a workflow which has been used in several practical applications [17] [18]. Interested readers are referred to [17] for more details. The first task creates a widget for users to interactively upload script files to be executed from local clients to the remote resource. The second task is to enable user running distributed copies of script using MPI [19]. The third step will help users to inspect the combined results.

The configuration file utilizes only three basic configuration options for each, task name, task types and description. Each of task type has a corresponding html view

with default appearance, control elements associated with specific server-side actions.

Figure 5. Example of workflow configuration and result web interface

In addition to three processing steps specified by configuration file, the application management component also generates control elements for users to interact with the application execution flow itself (Figure-5).

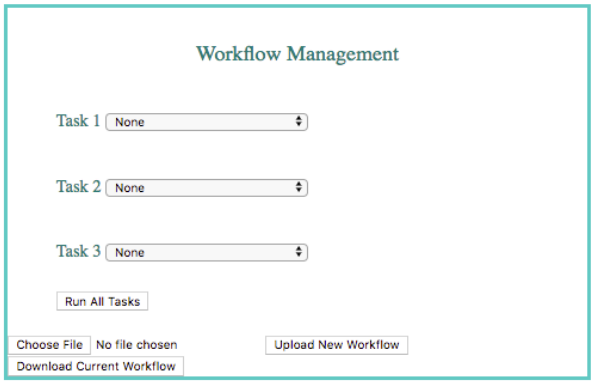

Figure 6. Workflow management interface.

There are three application interactions currently implemented: 1) download the current application as a JSON file; 2) upload a new application configuration file to change the application dynamically; 3) specify dependency among tasks. Using download and upload workflow features, a user can easily update a pre-defined workflow on demand. The capability of specifying dependency enables users to hold off execution of some tasks until some other tasks have finished. The entire application can run automatically in a given order without needs of user interaction. Tasks which do not depend on others or whose dependency has been met could be executed in parallel if there are enough resources.

E. Preliminary Implementation and Enviroments

We have implemented the proposed framework using Play Web Framework (version 2.6) [20]. Play framework is a Scala based modern web development framework. It has several features that especially suit for on-demand web application. It supports reactive web programming so that the content of web front can be dynamically updated along with the backend analysis progress. It includes a built-in lightweight web server, Jetty. This can greatly simplify the web UI deployment since it does not require any other additional system software and libraries other than Java. Play uses a template engine, Twirl, to generate web content. With template engine system, each web page is a result of a function call. Therefore, the content of web page is configurable through configuration files and the web page template files. Since the web page is in fact a Scala function, it can directly manipulate in memory data object with complex structure.

The authentication and authorization services are implemented using Silhouette authentication library (version 5.0). Silhouette supports several authentication methods, including OAuth1, OAuth2, OpenID, CAS, Credentials, Basic Authentication or custom authentication schemes [21].

For development and testing purpose, we used Wrangler cluster at Texas Advanced Computing Center as remote computing resource test-bed. Wrangler cluster has been designed to support the data storage and sharing capacities of the system to enable data research [22]. It supports dynamically Hadoop cluster provisioning, common data analysis, and machine learning software tools and libraries.

IV. Use Case Example

In this section, we demonstrate how the proposed web framework can be used to support and facilitate solutions for user driven tweets analysis.

With its fast communication and ease of publication, Twitter has become a massive and important social networking medium for people from all walks of life. Twitter has played a prominent role in influencing almost every aspect of everyday events, e.g. social-political events, such as the Occupy Wall Street movement and 2016 US presidential election; natural disasters, such as the Hurricane Sandy [23] [24]. Since Twitter's popularity as an information source has led to the development of applications and research in various domains, understanding the basics of collecting, and analyzing tweets has become an integral component of data science research and education [25].

Two major challenges in making use of Twitter data are the knowledge/skills to access real-time tweets and the resource to store and carry out large-scale computation. First, among a large number of tools for interacting with Twitter API to filter real-time tweets (i.e. filtering tweets by keywords as tweets are passing through the Twitter platform upon posting) and analyzing collected tweets, several provide easy access to the Twitter API, such as Tweepy for Python and rtweet for R. They are quite varied in their capabilities and require different levels of technical skills and infrastructure.

Second challenge lies in the large volume of twitter data. "big data" problems are not yet trained to take advantage of the data-intensive computing environments and relevant applications.

In practice, the following workflow has been used to gather live tweets, perform statistical analysis and sentimental analysis with collected tweets.

1. Login to the remote resources via ssh connection through a command line interface
2. Move libraries/tools required to the remote resource.
3. Submit a request to start collecting tweets using a Python script.
4. While the Tweets are being accumulated, an R script is used to perform statistical analysis on metadata of tweets, such as its origination locations.
5. After a number of tweets have been accumulated, sentimental analysis are conducted through a zeppelin notebook service.
6. Download the analysis results for inspection

Figure 7. A workflow requirements of collecting and analyzing tweets.

In this workflow, not only each step requires user intervention, tasks of steps 3 to 5 use different tools, programming environments and are started at different stages. On the other hand, the research group includes several students who are interested to carry out similar analysis but with different topics of interests. Running parameters of each step can change overtime depending on the specific research interest and users. These characteristics make it hard to generalize the workflow in a simple script and very difficult to be shared and reused among its potential users.

```
{
  "head":"Tweets Aggregationa and Analysis Workflow",
  "description":"test",
  "tasks":
    [
      {
        "task_name":"Upload files",
        "task_type":"fileUpload",
        "description":"upload prepare_files_and_directory.sh,streaming.py
          ,credentials.py,run_streaming_keywords.sh
          ,run_streaming_and_map_script.sh, process_tweets_log.R files"
      },
      {
        "task_name":"Run preparing script",
        "task_type":"runScript",
        "description":"In prepare_files_and_directory.sh, edit
          SOURCE_CODE_DIR to your upload directory, edit NEW_DIR to create a
          new directory to store required scripts and log folder"
      },
      {
        "task_name":"Run streaming script and map script",
        "task_type":"runScript",
        "description":"In run_streaming_and_map_script.sh, edit NEW_DIR to
          point to the new directory created "
      },
      {
        "task_name":"Show Result",
        "task_type":"showResult",
        "description":"Input tweets_map.png path and show tweets map"
      },
      {
        "task_name":"Hapdoop Reservation Information",
        "task_type":"checkHadoop",
        "description":"check Hapdoop reservation Information"
      },
      {
        "task_name":"Launch Zeppelin",
        "task_type":"startZeppelin",
        "description":"start Zeppelin server and load analysis notebook"
      }

    ]
}
```

Figure 8. Workflow of tweets collection and analysis as a Json file

The proposed web application framework overcomes those challenges by enable user defining the above workflow with a web application interface. After defining the tasks in the workflow (Figure 8), users can upload the workflow JSON file to formulate user-defined sequential steps for a real world problem on the web interface. Users can run all tasks with one click after specifying the dependency among the tasks (Figure 9).

The web application interface can also be shared among multiple users directly. The design of proposed framework allows users to add or remove individual pre-defined task via editing the workflow configuration file, so different users can also easily start different workflow interface.

The workflow shown in Figure 7 is now defined as five tasks in Figure 8. Figure 8 also includes an extra task to show cluster status.

The first task is a "fileupload" task through which a user can upload required files (in this case, five files may be required: a python script to authenticate with twitter API, a python script to streaming live tweets, a R scripts for statistical analysis, a bash scripts to prepare running environment, and a Zeppelin notebook for natural language processing). The interface for the step reduces the users' effort to understand the file transfer command 'scp' and provides a visual tree structure of remote file system (Figure 10).

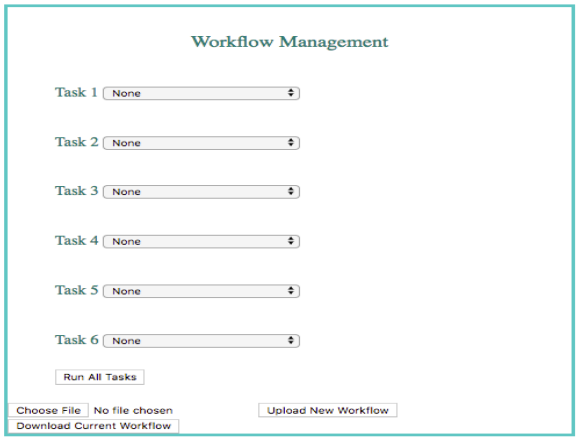

Figure 9. Interface of uploading workflow and defining dependency

Both second and third task are "run_script" types of tasks. One is for preparing the computing environment and the other is to start streaming live tweets. Although it is possible to merge two tasks into one, it is presented as two separate tasks for clarity and flexibility. The "run_script" task allow user to specify location of a bash script to be run on the remote resources. The task view also includes displaying content of the script to user directly. User can edit and change the script at the last minute on the fly. The changes can be saved in the remote resource to be reused in the future.

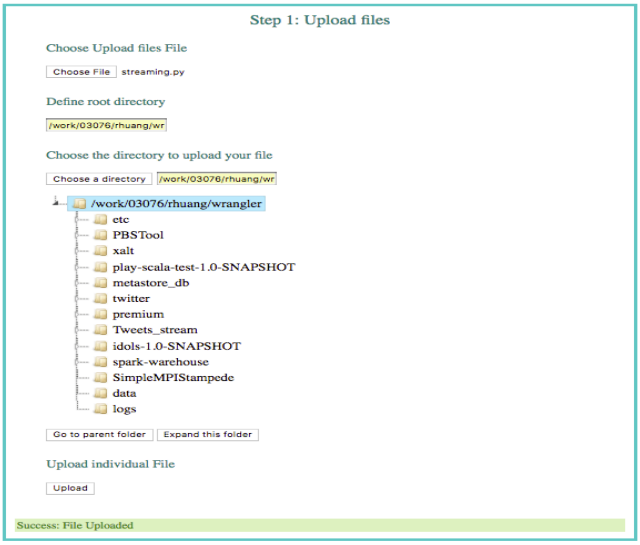

Figure 10. Interface of file upload from a user computer to data intensive computing resource

For step 3 using the same run script type of task as step 2, the streaming script has two arguments: keywords and stream time,. The keywords separated by comma are used to filter real-time tweets, e.g. with keywords "@WhiteHouse", "@realdonaldtrump" the application will collect tweets with "@WhiteHouse or @realdonaldtrump. The stream time define how long the real-time tweets collection lasts. The processing tweet log script will generate a tweets map with available locations from tweets.

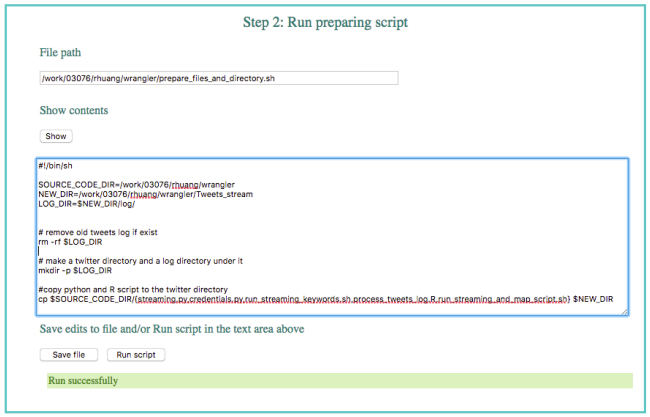

Figure 11. Interface of editing text file and run the script

The backend program to collect real-time tweets is adopted from an open source python code (https://github.com/zbeaver4/twitter_aws). The R script imports the JSON format of tweets and calculates the center of the bounding box of the place associated with the tweet. One of the results of R analysis a geospatial map of volume of tweets. Step 4 enables user to see the results directly without the needs to download the map explicitly from the remote server.

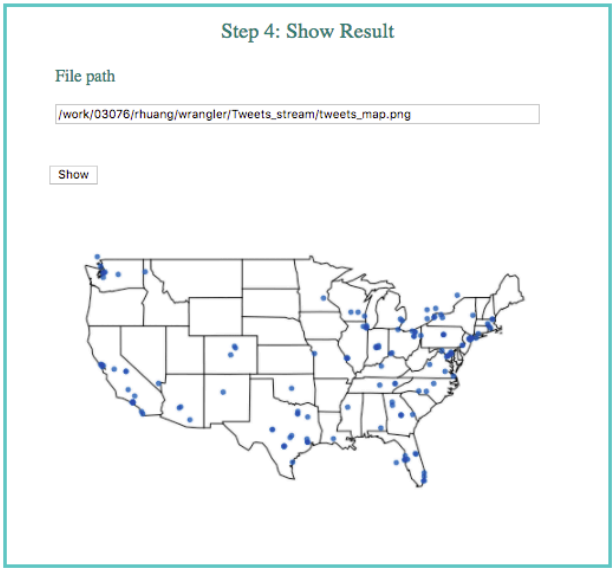

Figure 12. Interface to show the tweets map of live tweets.

To support further analysis over large volume of data, users can uses Spark and HDFS for efficient natural language processing. In this example, the analysis solution is in the form of a zeppelin notebook. Apache Zeppelin (https://zeppelin.apache.org/), a web-based notebook that enables data-driven, interactive data analytics and collaborative documents with SQL, Scala and more, can be launched on top of a Hadoop cluster. The step 6 will launch zeppelin service in the remote resources and load the corresponding analysis notebook. After successfully launch, a URL can be copied to web browser for access to Zeppelin web UI (Figure 13). Step 5 is an optional task which allows users to check the status of underline Hadoop cluster for zeppelin. The analysis notebook utilizing Spark program can be developed interactively in Zeppelin web UI for various types of analysis such as Linear Discriminant Analysis (LDA) and sentiment analysis (Figure 14).

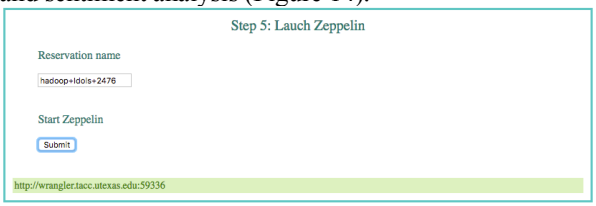

Figure 13. Interface of launching Apache Zeppelin on Hadoop cluster

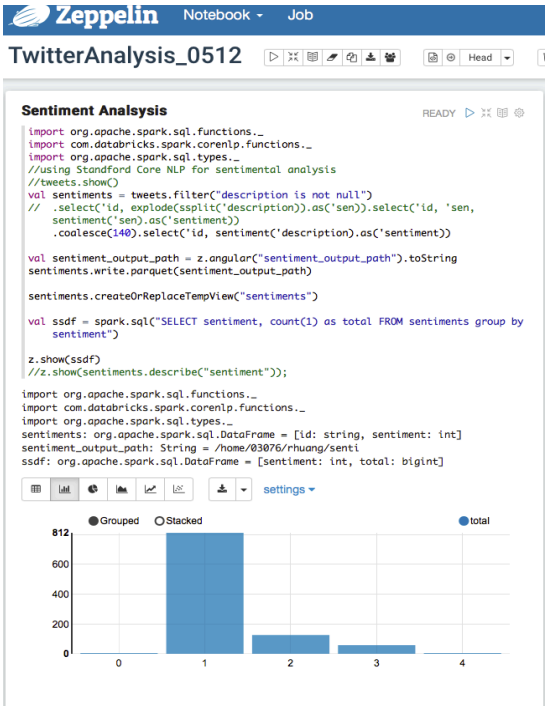

Figure 14. An example of sentiment analysis on Tweets using Zeppelin

V. Summary and Ongoing Work

In this paper, we have presented a web application framework proposal. A key motivation of this project is to use the framework to enhance the accessibility of large cyberinfrastructure to users from diverse domain fields. The framework enables cyberinfrastructure users to setup their own web services easily. The framework includes a set of pre-built task modules to help bridging users with remote hardware and software resources. By specifying JSON formatted configuration files, users can transform an ad-hoc analytic workflow into a dynamically composed multi-user interactive web application running on remote resources. The generated web application is self-contained with minimum system dependency on remote system. The composed workflow can also be exported, preserved and shared by other users. Using preliminary implementation, we demonstrate how the framework can simplify a workflow of aggregating and analysis tweets.

Our ongoing work includes extending the usability and extensibility of the framework by implementing additional re-usable task modules and supports of more remote resources and analysis tools. A specific aim is to use the framework as a foundation for training and education activities of cloud computing. Educators and instructors can easily compose, stage and publish interactive web session on remote resources. The framework can be used as a way to set up cloud based virtual laboratory for training and education on advanced computing technology.

Acknowledgment

This work has been supported by funding from National Science Foundation (Award# 1726816). Software testing and demonstrations have been supported with Wrangler, an NSF funded cyberinfrastructure resource (Award # 1341711).

References

[1] Gordon Bell, "Foreword," in *The Fourth Paradigm: Data-Intensive Scientific Discovery*, Tony Hey, Stewart Tansley, and Kristin Michele Tolle, Eds. Redmond, MA: Microsoft Research, 2009, pp. Xi-XV.

[2] Tom White, *Hadoop: The definitive guide.*: O'Reilly Media, 2012.

[3] Weijia Xu, Ruizhu Huang, Hui Zhang, David Walling, and Yaakoub El-Khamra, "Empowering R with High Performance Computing Resources for Big Data Analytics information," in *Conquering Big Data with High Performance Computing*, R. Arora, Ed.: Springer., 2016, pp. 191-218.

[4] Matei Zaharia et al., "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, 2012, p. 2.

[5] Merchant, Nirav, et al., "The iPlant Collaborative: Cyberinfrastructure for Enabling Data to Discovery for the Life Sciences," *PLOS Biology*, 2016.

[6] Stephen A. Goff et al., "The iPlant collaborative: cyberinfrastructure for plant biology," *Frontiers in plant science*, no. 2, p. 34, 2011.

[7] E.M. Rathje et al., "DesignSafe: new cyberinfrastructure for natural hazards engineering.," *Natural Hazards Review*, vol. 18, no. 3, 2017.

[8] Avita Katal and Mohammad Wazid and R. H. Goudar, "Big data: issues, challenges, tools and good practices," in *In Contemporary Computing (IC3), 2013 Sixth International Conference on.*, 2013, pp. 404-409.

[9] A. Yoo, M. Jette, and M. Grondona, "Slurm: Simple Linux Utility for Resource Management ," *Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science*, vol. 2862, pp. 44-60, 2003.

[10] Tristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood, and Andy Hopper, "Virtual network computing," *IEEE Internet Computing*, vol. 2, no. 1, pp. 33-38, 1998.

[11] Rstuido Team. R Studio. [Online]. https://www.rstudio.com/

[12] Diimitrios Georgakopoulos and Mark Hornick and Amit Sheth, ""An overview of workflow management: From process modeling to workflow automation infrastructure," *Distributed and parallel Databases*, vol. 3, no. 2, pp. 119-153, 1995.

[13] Suresh Marru et al., "Apache airavata: a framework for distributed applications and computational workflows," in *In Proceedings of the 2011 ACM workshop on Gateway computing environments*, 2011, pp. 21-28.

[14] Ilkay Altintas, Chad Berkley, Efrat Jaeger, Matthew Jones, and Bertram Ludascher and Steve Mock, "Kepler: an extensible system for design and execution of scientific workflows," in *16th Scientific and Statistical Database Management 2004* , 2004, pp. 423-424.

[15] Malcolm Atkinson, Sandra Gesing, Johan Montagnat, and Ian Taylor, "Scientific workflows: Past, present and future,"

Future Generation Computer Systems, no. 75, pp. 216-227, 2017.

[16] Avraham and James T. Rayfield Leff, "Web-application development using the model/view/controller design pattern," in *Fifth IEEE International Enterprise Distributed Object Computing Conference (EDOC'01)*, 2001, pp. 118-127.

[17] Yu Qian et al., "FlowGate: towards extensible and scalable web-based flow cytometry data analysis.," in *In Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure (XSEDE '15)*, St. Louis, MO, USA, 2015, p. 8.

[18] Jesse R. Lasky et al., "Natural variation in abiotic stress responsive gene expression and local adaptation to climate in Arabidopsis thaliana.," *Molecular biology and evolution*, vol. 31, no. 9, pp. 2283-2296, 2014.

[19] William Gropp, Ewing Lusk, Nathan Doss, and Anthony Skjellum, "A high-performance, portable implementation of the MPI message passing interface standard," *Parallel computing*, vol. 22, no. 6, pp. 789-828, 1996.

[20] Play Development Team. Play Web Framework. [Online]. https://www.playframework.com/

[21] Silhouette. Silhouette authentication library. [Online]. https://www.silhouette.rocks/

[22] Christopher, David Walling, Weijia Xu, Stephen A. Mock, Niall Gaffney, and Dan Stanzione Jordan, "Wrangler's user environment: A software framework for management of data-intensive computing system," in *2015 IEEE International Conference on Big Data*, 2015, pp. 2479-2486.

[23] Shamanth Kumar and Fred Morstatter and Huan Liu, *Twitter data analytics*.. New York, USA, 2014.

[24] "Fohringer, J., D. Dransch, H. Kreibich, and K. Schröter. "Social media as an information source for rapid flood inundation mapping," *Natural Hazards and Earth System Sciences* , vol. 15, no. 12, pp. 2725-2738, 2015.

[25] Oshini Goonetilleke, Timos Sellis, and Xiuzhen Zhang and Saket Sathe, "Twitter analytics: a big data management perspective," *ACM SIGKDD Explorations Newsletter*, vol. 16, no. 1, pp. 11-20, 2014.