Devising a Cloud Scientific Workflow Platform for Big Data

Yong Zhao, Youfu Li
School of Computer Science and Engineering
Univ. of Electronic and Science Technology of China
Chengdu, China
{yongzh04, youfuli.fly}@gmail.com

Shiyong Lu
Department of Computer Science
Wayne State University
Detroit, USA
shiyong@wayne.edu

Ioan Raicu
Department of Computer Science
Illinois Institute of Technology
Chicago, USA
iraicu@iit.edu

Cui Lin
Department of Computer Science
California State University
Fresno, USA
clin@csufresno.edu

***Abstract*—Scientific workflow management systems (SWFMSs) are facing unprecedented challenges from big data deluge. As revising all the existing workflow applications to fit into Cloud computing paradigm is impractical, thus migrating SWFMSs into the Cloud to leverage the functionalities of both Cloud computing and SWFMSs may provide a viable approach to big data processing. In this paper, we first discuss the challenges for scientific workflow applications and the available solutions in details, and analyze the essential requirements for a scientific computing Cloud platform. Then we propose a service framework to normalize the integration of SWFMS with Cloud computing. Meanwhile, we also present our implementation experience based on the service Framework. At last, we set up a series of experiments to demonstrate the capability of our implementation and use a Montage Image Mosaic Workflow as a showcase of the implementation.**

Keywords—Big Data; Cloud Scientific Workflow Management; Service Framework; Interface Definition

I. INTRODUCTION

Scientific workflow management systems (SWFMSs) have been widely adopted in Physics, Astronomy, Bioinformatics, Neuroscience, Earth Science, and Social Science to provide essential support to scientific computing, including management of data and task dependencies, job scheduling and execution, provenance tracking, etc. Nevertheless, the big data era has raised unprecedented challenges against the solution to data processing based on traditional scientific workflows, as the data scale and computation complexity are growing exponentially. The ETL (Extraction-Transformation-Loading), storage, retrieval, analysis and application upon the huge amounts of data are beyond the capability of traditional data processing infrastructures.

As an emerging computing paradigm, Cloud computing[6] has brought us tremendous convenience for the processing of large-scale datasets. Hadoop[22], together with associated systems in its ecosystem, has been widely adopted to solve the big data problem arisen in both scientific community and enterprises, as they provide a scalable and large-scale solution to data storage, analysis and mining. Theoretically, to address the big data problems in these areas, scientists and application developers may simply refactor all the existed workflow applications into the Cloud computing paradigm, which looks straightforward but in reality impractical. As traditional scientific workflow applications have been mature during many years' development and always involve complicated application logic and consist of massive computing processes, including organization, distribution, coordination and parallel processing. Transforming these scientific workflows will not only cost scientists and developers much time, but also manually handle all the integration details with various underlying Cloud platforms.

An alternative for researchers is to integrate scientific workflow management systems with Clouds, leveraging the functionalities of both Cloud computing and SWFMSs to provide a Cloud workflow platform as a service for big data processing. In this solution, not only the challenges for traditional scientific workflows can be dealt with, but also the researchers can concentrate on applications and utilize the integration platform to process massive data on Clouds. As workflow management systems are diverse in many aspects, such as workflow models, workflow languages, workflow engines, and so on, and each workflow system engine may depend on one specific Distributed Computing Infrastructures (DCIs), porting a workflow management system to run on another DCI may cost a large quantity of extra effort. We would like to free researchers from complicated integration details, such as Cloud resource provisioning, task scheduling and so on, and provide them with the convenience and transparency to scalable big data processing platform, therefore we propose a service framework to standardize the integration between SWFMSs and Cloud platforms, breaking the limitations that a specific SWFMS is bound to a particular DCI or Cloud environment. We define a series of components and interfaces to normalize the interactions between different workflow management subsystems.

II. Related Work

Systems such as Taverna [11], Kepler [9], Vistrails [10], Pegasus [8], Swift [20], and VIEW [25] have seen wide adoption in various disciplines such as Physics, Astronomy, Bioinformatics, Neuroscience, Earth Science, and Social Science. In Table I, we list some use cases that focused on applying SWFMSs to execute data-intensive applications.

TABLE I. Use Cases of SWFMSs

SWFMSs	Application Fields	Use Cases
Swift	Climate Science	Climate Data Analysis[1]
Taverna	Bioinformatics	Single Nucleotide Polymorphisms Analysis[2]
Vistrails	Earth Science	NASA Earth Exchange [3]
Kepler	Physics	Hyperspectral image processing [4]
VIEW	Medical Science	Neurological disorder diagnosis[13]

There are also some researchers that tried to run workflow applications on Clouds. The series of works [12][18] focused on running scientific workflows that are composed of loosely coupled parallel applications on various Clouds. The study conducted on an experimental Nimbus Cloud testbed [17] dedicated to science applications involved a non-trivial amount of computation performed over many days, which allowed the evaluation of the scalability as well as the performance and stability of the Cloud over time. Their studies demonstrated that the multi-site Cloud computing is a viable and effective solution for some scientific workflows, and the networking and management overhead across different Cloud infrastructures do not have a major effect on the overall user experience, and the convenience of being able to scale resources at runtime outweighs such overhead.

The deployment and management of workflows over the current existing heterogeneous and not yet interoperable Cloud providers, however, is still a challenging task for the workflow developers. The series of works [14] [16] presented a broker-based framework to support the execution of workflow applications on a multi-Cloud environment.

The CODA framework [23] was designed and implemented to support big data analytics in Cloud computing. Important functions, such as workflow scheduling, data locality, resource provisioning, and monitoring functions, had been integrated into the framework. Through the CODA framework, the workflows could be easily composed and efficiently executed in Amazon EC2. In order to address performance and cost issues of big data processing on Clouds, Long Wang et al. [15] presented a novel design of adaptive workflow management system which included a data mining based prediction model, workflow scheduler, and iteration controls to optimize the data processing via iterative workflow tasks.

Those works mentioned above were mainly focused on applications of SWFMSs and different aspects of the deployment and management of integrating workflows into Clouds, including underlying resource allocation, function implementation, service evaluation, performance and cost issues, etc., however, a normalized, service-oriented integration framework is still missing. As running scientific workflows as a service in the Cloud platforms involves a variety of systems and techniques, defining the pivotal interfaces can help to normalize the interactions between essential systems.

III. Challenges and Available Solutions

In this section, we discuss the challenges of utilizing traditional scientific workflows to deal with big data problems and analyze the available solutions to the following challenges.

A. Challenges for Tranditional Scientific Workflows

Scientific workflow systems have been formerly applied over a number of execution environments such as workstations, clusters/Grids, and supercomputers. In contrast to Cloud environment, running workflows in these environments are facing a series of obstacles when dealing with big data problems [5], including data scale and computation complexity, resource provisioning, collaboration in heterogeneous environments, etc..

1) Data Scale and Computation Complexity

The execution of scientific workflows often consume and produce huge amounts of distributed data objects. These data objects can be of primitive or complex types, files in different sizes and formats, database tables, or data objects in other forms. At present, the scientific community is facing a "data deluge" [7] coming from experiments, simulations, networks, sensors, and satellites, and the data that needs to be processed generally grows faster than computational resources and their speed. The data scale and management in big data era are beyond the capability of traditional workflows can handle as they depend on traditional infrastructure for resource provisioning. For example, in high energy physics, the Large Hadron Collider[1] at CERN can generate more than 100TB of collision data per second; In bioinformatics, GenBank[2], one of the largest DNA databases, already hosts over 120 billion bases, the European Molecular Biology and Bioinformatics Institute Laboratory (EMBL) hosts 14 PB of data, and the numbers are expected to double every 9-12 months.

In addition to data scale, science analysis and processing complexity is also growing exponentially. Scientists are now attempting calculations requiring orders of magnitude more computing and communication than was possible only a few years ago. For instance, in bioinformatics a protein simulation problem [19] involves running many instances of a structure prediction simulation, each with different random initial conditions and performs multiple rounds. Given a couple of proteins and parameter options, the simulation can easily scale up to 100,000 rounds. In cancer drug design, protein docking can involve millions of 3D structures and have a runtime up to tens of CPU years. To enable the storage and analysis of such large quantities of data and to achieve rapid turnaround, data and computation may need to be distributed over thousands or even tens of thousands of computation nodes.

2) Resource Provisioning

Resource provisioning represents the functionality and mechanism of allocating computing resource, storage space, network bandwidth, etc., to scientific workflows. As cluster/Grid environments is not adept at providing the

[1] http://lhc.web.cern.ch
[2] http://www.ncbi.nlm.nih.gov/genbank

workflows with smoothly dynamic resource allocation, the resource provisioned to a scientific workflow is fixed once the workflow has been deployed to execute, which may in return restrict the scale of science problems that can be handled by workflows. Moreover, the scale of resource is upbounded by the size of a dedicated resource pool with limited resource sharing extension in the form of virtual organizations. Meanwhile, the representation of resources in the context of scientific workflows is also bothering the scientists [28], as they must be able to recognize the supported types of resources and tools. For instance, the resource in Taverna is a web service which usually limits the use of many scientific resources that are not represented as web services.

To break through the limitations introduced by traditional resource provisioning strategy, some works have been focused on the approaches for automated provisioning, including the Context Broker [24] from the Nimbus project, which supported the concept of "one-click virtual cluster" that allowed clients to coordinate large virtual cluster launches in simple steps. The Wrangler system [26] was a similar implementation that allowed users to describe a desired virtual cluster in XML format, and send it to a web service, which managed the provisioning of virtual machines and the deployment of software and services. It was also capable of interfacing with many different Cloud resource providers.

3) Collaboration in Heterogeneous Environments

Collaboration refers to the interactions between workflow management system and execution environment, such as resource access, resource status perception, load balance and so on. As more and more scientific research projects become collaborative in nature and involve multiple geographically distributed organizations, which brings a variety of challenges to scientists and application developers to handle the collaboration in heterogeneous environments.

The management of resource, authority authentication, security, etc., can be very complicated, as scientific workflow applications are normally executed in cluster/Grid environments, where accessible computing resources and storage space are located in various management domains. The execution of traditional workflows are also influenced by the heterogeneous performance of computing resource due to the varied configuration of physical machines. In addition, in Grid environment, the status of physical machines is uncontrollable, switching among online (the machine is started up and connected to the Grid), offline (the machine is powered off or disconnected), busy (the machine is executing other tasks), etc., making it extremely difficult to maintain load balance.

B. Moving Workflow Applications to Cloud

Since Cloud computing has been widely adopted to solve the ever-increasing computing and storage problems arising in the Internet age. To address the challenges of dealing with peta-scale scientific problems in scientific workflow solutions, we can move workflow applications into Cloud, using the MapReduce computing model to reconstruct the formerly applied workflow specifications. MapReduce provides a very simple programming model and powerful runtime system for the processing of large datasets. The programming model is based on just two key functions: "map" and "reduce," borrowed from functional languages. The runtime system automatically partitions input data and schedules the execution of programs in a large cluster of commodity machines. Modified applications to fully leverage the unprecedented scalability and resources on demand offered by the Cloud without introducing extra management overheads.

Despite all the advantages of transforming traditional workflow applications into Cloud-based applications, there are still some shortcomings and unsolved obstacles:

1) Cloud computing cannot benefit from the distinguishing features provided by SWFMSs, including management of data and task dependencies, job scheduling and execution, provenance tracking, etc.. The challenges for big data processing in Cloud remain unsolved and are still bothering developers and researchers.

2) Utilizing the certain data flow support offered by MapReduce to refactor traditional workflow applications require application logic to be rewritten to follow the map-reduce-merge programming model. Scientists and application developers need to fully understand the applications and port the applications before they can leverage the parallel computing infrastructure.

3) Large-scale workflows, especially data-intensive scientific workflows [29] may require far more functionality and flexibility than MapReduce can provide, and the implicit semantics incurred by a workflow specification goes far more than just the "map" and "reduce" operations, for instance, the mapping of computation to compute node and data partitions, runtime optimization, retry on error, smart re-run, etc.

4) Once we decide to migrate workflow applications to Cloud computing, we need to reconstruct the data being processed to be able to be stored in partitioned fashion, such as in GFS, or HDFS, so that the partitions can be operated in parallel, which may introduce a tremendous amount of work to scientists and application developers.

5) Revising workflow applications to be capable of executing in Cloud platforms makes new requests to scientists and application developers, as they need to grasp new programing model and techniques instead of using already-familiar workflow pattern, which may cost large amount of time beyond the research topics. Moreover, the risks associated with vendor lock-in cannot be ignored.

C. Migrating Workflow Management into Cloud

To avoid the disadvantages brought by moving workflow applications directly to Cloud, we may try to integrate workflow management systems with Cloud to provide a Cloud workflow platform as a service for big data processing. Once we decide to integrate SWFMS with Cloud computing, we may deploy the whole SWFMS inside the Cloud and access the scientific workflow computation via a Web browser. A distinct feature of this solution is that no software installation is needed for a scientist and the SWFMS can fully take advantage of all the services provided in a Cloud infrastructure. Moreover, the Cloud-based SWFMS can provide highly scalable scientific workflows and task management as services, providing one kind of Software-as-a-Service(SaaS). One concern the user

might have is the economic cost associated with the necessity of using Cloud on a daily basis, the dependency on the availability and reliability of the Cloud, as well as the risk associated with vendor lock-in.

To provide a good balance between system performance and usability, an alternative for researchers is to encapsulate the management of computation, data, and storage and other resources into the Cloud, while the workflow specification, submission, presentation and visualization remain outside the Cloud to support the key architectural requirement of user interface customizability and user interaction support. The benefit of adopting the solution to manage and run scientific workflows on top of the Cloud can be multifold:

1) The scale of scientific problems that can be addressed by scientific workflows can be greatly increased compared to cluster/Grid environments, which was previously upbounded by the size of a dedicated resource pool with limited resource sharing extension in the form of virtual organizations. Cloud platforms can offer vast amount of computing resources as well as storage space for such applications, allowing scientific discoveries to be carried out in a much larger scale.

2) Application deployment can be made flexible and convenient. With bare-metal physical servers, it is not easy to change the application deployment and the underlying supporting platform. However with virtualization technology in a Cloud platform, different application environments can be either pre-loaded in virtual machine (VM) images, or deployed dynamically onto VM instances.

3) The on-demand resource allocation mechanism in the Cloud can improve resource utilization and change the experience of end users for improved responsiveness. Cloud-based workflow applications can get resources allocated according to the number of nodes at each workflow stage, instead of reserving a fixed number of resources upfront. Cloud workflows can scale out and in dynamically, resulting in fast turn-around time for end users.

4) Cloud computing provides much larger room for the trade-off between performance and cost. The spectrum of resource investment now ranges from dedicated private resources, a hybrid resource pool combining local resource and remote Clouds, and full outsourcing of computing and storage to public Clouds. Cloud computing not only provides the potential of solving larger-scale scientific problems, but also brings the opportunity to improve the performance/cost ratio.

5) Although migrating scientific workflow management to Cloud may introduce extra management overheads, Cloud computing now can leverage the advantages carried about with SWFMSs (e.g. workflow management, provenance tracking, etc.).

IV. Requirements, Design and Implementation

In this section, we present our structured approach to designing and deploying a Cloud workflow management platform for big data processing. We first analyze the requirements for such a platform from the perspective of scientists and researchers, we then discuss our service framework and analyze the implementation in details from the perspective of interfaces, which standardize the interactions between associated subsystems.

A. Design Requirements

From the perspective of scientists and researchers, a Cloud platform for scientific computing should cover the end-to-end application execution scenario, from job specification, job submission, resource provisioning, to job execution and result delivery. We argue that a scientific computing Cloud platform should meet the following requirements:

Application development environment: Scientists and application developers would need a development environment for the specification, debugging and testing of their application logic, and the environment could also support easy deployment and execution of the application in the Cloud platform. One example is application developers for Windows Azure Services from Microsoft could use Visual Studio to develop their Cloud applications, test and run the applications locally, and then deploy to the Azure Cloud platform. Having such an environment would greatly facilitate the development process and simplify interfacing to the Cloud.

Cloud gateway service: Just like the way scientists interact with a Grid computing platform such as TeraGrid [21], they would need a similar interface in the form of a gateway or portal such that they can access resources in the back and run their applications without becoming experts on Grid or Cloud technologies. In addition to servicing job submissions and resource requests, the gateway can also manage user authentication and authorization, and keep track of resource usage and accounting related issues.

Virtual Resource provisioning: One of the major benefits of Cloud is its resources on demand. With the right resource provisioning mechanism, applications running on the Cloud can easily scale out and in, and achieve good cost performance balance. Combined with virtualization technology, virtual resource provisioning can provide science applications with not only the scalability to tap into the potentially unlimited resources, but also the flexibility to set up and deploy the necessary infrastructure and environment required for the applications to run. Virtual cluster provisioning can even provision a cluster of virtual machine instances with the network and storage all set up, and in some implementations a ready-to-use scheduler such as PBS for job scheduling.

Job scheduling and execution: As we have pointed out before, getting the necessary resources for the application is not the end of story to running an application. We would also need to schedule and coordinate the execution of the jobs in the application, and balance resource utilization across the jobs. A job scheduling and execution service needs to be deployed on top of the acquired virtual resources for efficient execution.

Infrastructure level support: An underlying Cloud platform should provide the basic infrastructure level support such as management of physical servers, network and storage devices, configuration and instantiation of virtual machines, as well as virtual image management and logging etc. In addition to the ones listed above, there are other functional requirements, such as monitoring and error recovery of job execution, result visualization, provenance tracking, etc. to a

science computing Cloud platform. There are also performance related issues, such as the instantiation time of virtual machine instances and clusters, resource scheduling efficiency, and so on.

B. The Service Framework

We propose a reference service framework that fulfills the above design requirements and covers all the major aspects involved in the migration and integration of SWFMS into the Cloud, from client-side workflow specification, service-based workflow submission and management, task scheduling and execution, to Cloud resource management and provisioning. As illustrated in Fig. 1, the service framework includes 4 layers, 8 components and 6 interfaces. Detailed description of the service framework is made public at our website[3].

The first layer is the Infrastructure Layer, which consists of multiple Cloud platforms with the underlying server, storage and network resources. The second layer is called the Middleware Layer. This layer consists of three subsystems: Cloud Resource Manager, Scheduling Management Service and Task Scheduling Frameworks. The third layer, called the Service Layer, consists of Cloud Workflow Management Service and Workflow Engines. Finally, the fourth layer – the Client Layer, consists of the Workflow Specification & Submission and the Workflow Presentation & Visualization subsystem. The service framework would help to break through workflows' dependence on the underlying resource environment, and take advantage of the scalability and on-demand resource allocation of the Cloud.

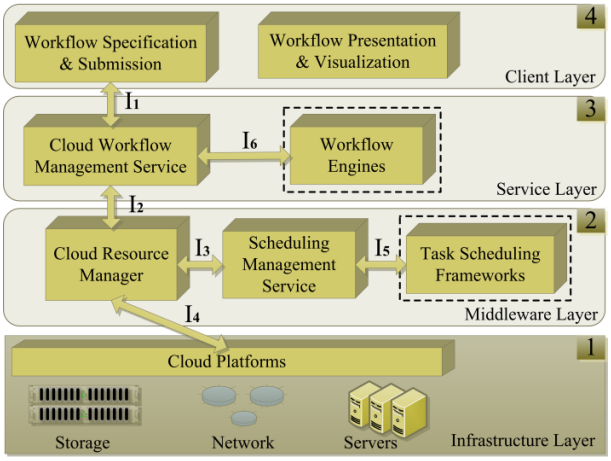

Fig. 1. The Service Framework

We present a layered service framework for the implementation and application of integrating SWFMS into manifold Cloud platforms, which can also be applicable when deploying a workflow system in Grid environments. The separation of each layer enables abstractions and different independent implementations for each layer, and provides the opportunity for scientists to develop a stable and familiar problem solving environment where rapid technologies can be leveraged but the details of which are shielded transparently from the scientists who need to focus on science itself. The Interfaces defined in the framework is flexible and customizable for scientists to expand or modify according to their own specified requirements and environments.

[3] http://www.cloud-uestc.cn/projects/serviceframework/index.html

C. Implementation

We also implement the service framework by integrating the Swift scientific workflow management system [20] with the OpenNebula and Eucalyptus platforms. The integration supports workflow specification and submission, on-demand virtual cluster provisioning, high-throughput task scheduling and execution, and scalable resource management in the Cloud. The layers, systems and interfaces displayed in the integration architecture can be easily mapped into the corresponding components in the service framework. As we can choose different available systems for the integration, we would like to focus on the interactions between the associated subsystems when interpreting the implementation.

1) Associated Subsystems

As the implementation of service framework includes a variety of systems and techniques, for the purpose of clarity, we list the subsystems, corresponding to Fig. 1, in table II. And we also point out which subsystems are directly from the original systems and which are implemented for the integration.

TABLE II. SUBSYSTEMS IMPLEMENTATION DESCRIPTION

Subsystems	Description	Mapped Subsystems
OpenNebula/Eucalyptus	reuse	Cloud Platforms
Falkon Scheduling Framework [27]	minor revision	Task Scheduling Frameworks
Scheduling Management Service	implemented	Scheduling Management Service ***(Abbr. SMS)***
Cloud Resource Manager	implemented	Cloud Resource Manager
Swift System	minor revision	Workflow Engines
Cloud Workflow Management Service	implemented	Cloud Workflow Management Service ***(Abbr. CWMS)***
Client Submission Tool	implemented	Workflow Specification & Submission

[a.] "reuse": we directly reuse the available components for integration

[b.] "minor revision": we reuse the available components after customization.

[c.] "implemented": we implement the components from design to test.

The Swift Workflow Management System

Swift is a system that bridges scientific workflows with parallel computing. Swift takes a structured approach to workflow specification, scheduling, and execution. It consists of a simple scripting language called SwiftScript for concise specification of complex parallel computations based on dataset typing and iterations, and dynamic dataset mappings for accessing large-scale datasets represented in diverse data formats.

The Swift system architecture consists of four major components: Program Specification, Scheduling, Execution, and Provisioning. Computations are specified in SwiftScript, which has been shown to be simple yet powerful. SwiftScript programs are compiled into abstract computation plans, which are then scheduled for execution by the workflow engine onto

provisioned resources. Resource provisioning in Swift is very flexible, tasks can be scheduled to execute on various resource providers, where the provider interface can be implemented as a local host, a multi-site Grid, or the Amazon EC2 service.

TABLE III. INTERFACES IMPLEMENTATION DESCRIPTION

Interfaces	Description	Mapped Into	Interaction Between
IWorkflowSubmission	implemented	Interface I_1	Workflow Specification & Submission and ***CWMS***
IVirtualClusterRequest	implemented	Interface I_2	***CWMS*** and Cloud Resource Manager
ISchedulingFrameworkDeployment	implemented	Interface I_3	Cloud Resource Manager and ***SMS***
IVirtualMachineOperation	implemented	Interface I_4	Cloud Resource Manager and Cloud Platforms
ISchedulingFrameworkOperation	under evaluation	Interface I_5	***SMS*** and Task Scheduling Frameworks
IWorkflowJobSubmission	under evaluation	Interface I_6	***CWMS*** and Workflow Engines

[a.] "implemented": means we define and implement the interfaces.

[b.] "under evaluation": represents those interfaces have been defined but still need further adjustment and evaluation for detail implementation.

2) Interface Definitions

The realization of service framework also involves complicated communications between different essential subsystems. These interfaces, not bound to one specific system, are defined to normalize the interactions between associated systems. Available systems can be easily integrated into the service framework as long as they can be modified to realize the corresponding interfaces. For instance, we can deploy CloudStack as the underlying Cloud platform as long as we encapsulate the CloudStack's API according to the interfaces between the Cloud Resource Manager and Cloud Platforms.

Corresponding to Fig. 1, we list the key interfaces in Table III, and point out the implementation status and interaction relationships. We will introduce the definitions of some pivotal interfaces in the following, and further details about these interfaces are available at our website[4].

In Table IV, we present the definition of interface "IWorkflowSubmission", which can be mapped into Interface I_1. We define the interface "IWorkflowSubmission" to standardize the submission of workflows from the Workflow Specification & Submission to Cloud Workflow Management Service.

TABLE IV. DEFINITION OF IWORKFLOWSUBMISSION

```
public interface IWorkflowSubmission {
    public boolean submitWorkflow(WorkflowSpecification workflow,
ExecutionConfiguration config) throws Exception;
    public WorkflowStatus queryWorkflowStatus(String workflowID)
throws Exception;
    public WorkflowResult queryWorkflowResult(String workflowID)
throws Exception;
    public boolean retractWorkflowSubmission(String workflowID)
throws Exception;
…}
```

In Table V, we list a set of operations corresponding to Interface I_2, for the interactions between Cloud Workflow Management Service and Cloud Resource Manager, such as sending a cluster request, querying cluster information and releasing a cluster after execution.

TABLE V. DEFINITION OF IVIRTUALCLUSTERREQUEST

```
public interface IVirtualClusterRequest {
    public VirtualCluster requestCluster(int clusterSize, ClusterDetails
details) throws Exception;
    public VirtualCluster queryClusterInformation(String clusterID)
throws Exception;
    public boolean releaseCluster(String clusterID) throws Exception;
…}
```

We define the interface "ISchedulingFramework-Deployment" (displayed in Table VI), which can be mapped into Interface I_3, to standardize the deployment of scheduling framework with specified deployment configuration. Other operations, associated with the deployment of Scheduling Management Service, are also defined in this interface.

TABLE VI. DEFINITION OF ISCHEDULINGFRAMEWORKDEPLOYMENT

```
public interface ISchedulingFrameworkDeployment {
    public SchedulingFrameworkInformation deployScheduling-
Framework(VirtualCluster virtualCluster,
SchedulingFrameworkConfiguration configuration) throws Exception;
    public SchedulingFrameworkInformation modifyScheduling-
Framework(VirtualCluster virtualCluster, SchedulingFramework-
Information information, SchedulingFrameworkConfiguration
configuration) throws Exception;
    public boolean revokeSchedulingFrameworkDeployment
(VirtualCluster virtualCluster, SchedulingFrameworkInformation
information) throws Exception;
…}
```

The definition of interface "IVirtualMachineOperation" in Table VII can be mapped into Interface I_4. This interface covers the general operations upon virtual machines, including instance creation, launching, shutdown, reboot, etc., which can be the encapsulation standard of Cloud platforms API.

TABLE VII. DEFINITION OF IVIRTUALMACHINEOPERATION

```
public interface IVirtualMachineOperation {
    public ArrayList<VirtualMachineInformation> createInstances (int
vmNumber) throws Exception;
    public ArrayList<VirtualMachineInformation> launchInstances
(ArrayList <VirtualMachineInformation> vmList) throws Exception;
    public boolean shutdownInstances (ArrayList <VirtualMachine-
Information> vmList) throws Exception;
    public ArrayList<VirtualMachineInformation> rebootInstances
(ArrayList <VirtualMachineInformation> vmList) throws Exception;
    public boolean destroyInstances (ArrayList <VirtualMachine-
Information> vmList) throws Exception;
…}
```

To normalize the submission of workflow job from Cloud Workflow Management Service to Workflow Engines, we define a series of operations in "IWorkflowJobSubmission" (as shown in Table VIII), corresponding to Interface I_6. Covering the general operations upon submitting workflow job to Workflow Engines.

TABLE VIII. DEFINITION OF IWORKFLOWJOBSUBMISSION

```
public interface IWorkflowJobSubmission {
    public boolean submitWorkflowJob(WorkflowSpecification
```

[4] http://www.cloud-uestc.cn/projects/serviceframework/index.html.

```
specification, VirtualCluster cluster, WorkflowEngine engine) throws Exception;
    public WorkflowStatus checkWorklfowExecutionStauts (String workflowID) throws Exception;
    public WorkflowResult fetchWorkflowResult(String workflowID) throws Exception;
    public boolean cancelWorkflowExecution(String workflowID) throws Exception;
…}
```

V. EVALUATION

In this section, we show our experiment results of implementation for both the OpenNebula and Eucalyptus platforms to demonstrate the practicability and capability of our implementation.

A. Experiment Configuration

OpenNebula: We use 6 machines in the experiment, each configured with Intel Core i5 760 with 4 cores at 2.8GHZ, 4GB memory, 500GB HDD, and connected with Gigabit Ethernet LAN. The configuration for each VM is 1 core, 1.5GB memory, 20GB HDD, and we use KVM as the hypervisor. One of the machines is used as the frontend which hosts the workflow service, the CRM, and the monitoring service. The other 5 machines are used to instantiate VMs, and each physical machine can host up to 2 VMs, so at most 10 VMs can be instantiated in the environment.

Eucalyptus: Considering the efficient and convenient service provided by the FutureGrid[5], we choose Eucalyptus for the implementation and deployment. FutureGrid is a project led by Indiana University and funded by the National Science Foundation (NSF) to develop a high-performance Grid test bed that lets scientists collaboratively develop and test innovative approaches to parallel, Grid, and Cloud computing. We measure the performance to establish a baseline for resource provisioning and Cloud resource management overhead in the science Cloud environment.

The instance type used in our experiment is m1.small: 1 CPU Unit, 1 CPU Core and 500MB Memory. All the instances use Ubuntu Server 12.04 as the operating system.

B. Resource Provisioning

In our implementation, we have realized the dynamic resource request by interacting with underlying Cloud platforms. Considering the experiments are conducted in the laboratory environment, where economic cost can be temporarily ignored, we pre-instantiate all the required VMs and put them in the VM pool, which may help the evaluation results be more intuitionistic and comparable.

1) The base line measurement

We first measure the base line for server initialization time and worker registration time. We create a Falkon virtual cluster with 1 server, and varying number of workers, and we don't reuse the virtual cluster.

The base line results in Fig. 2 are measured in OpenNebula environment. We can observe that the server initialization time is quite stable, around 4.7s every time, and for worker parallel registration, the time increases slightly with the worker number.

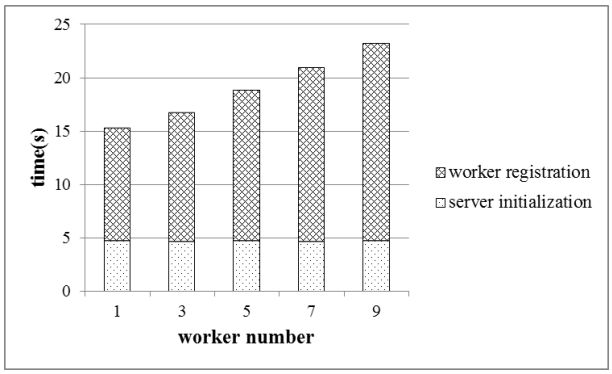

Fig. 2. The Base Line Measurement (OpenNebula)

We measure the server initialization time and worker registration time (illustrated in Fig. 3) in Eucalyptus environment to compare with those in the OpenNebula setting. We observe the time to create a Falkon server and start the service is around 11s, much longer than that in Fig.2. We attribute this to the m1.small configuration. The overall time increases slightly with the worker number as all the worker registration is executed concurrently, which shows a similar pattern to that in Fig. 2.

Fig. 3. The Base Line Measurement (Eucalyptus)

2) Performance improvement

We implement an optimization technique to speed up the cluster creation. When a Falkon virtual cluster is decommissioned, we change its status to "standby", and it can be re-activated. When the Cloud Resource Manager receives resource request, it checks if there is a "standby" Falkon cluster, if so, it will return the information of the Falkon service, and also checks the number of the Falkon workers already in the cluster.

In Eucalyptus environment, We measure the recycling mechanism by submitting requests with exponentially decreasing worker number. Except the first request, the server initialization time of the other requests is zero, and the time taken is to deregister 16 workers→8 workers→4 workers→2 workers→1 worker from previous "standby" Falkon cluster

We use (1) to calculate the percent of time saved when applying recycling mechanism for resource provisioning. The P_{TS} means the percent of time saved, and T_B represents the base

[5] FutureGrid: https://portal.futuregrid.org/

line to create a Falkon cluster with specified worker number. T_C indicates the time cost to initialize a Falkon cluster when applying the recycling mechanism. As shown in table IX, we can clearly see the recycling mechanism is efficient when initializing a cluster based on the "standby" Falkon cluster.

$$P_{TS} = \frac{T_B - T_C}{T_B} \times 100\% \quad (1)$$

TABLE IX. PERFORMANCE IMPROVEMENT (EUCALYPTUS)

Cluster Size	Time(s)	Base Line(s)	Time Saved(percent)
32	39.598	40.376	-
16	9.548	34.173	72%
8	7.61	28.331	73%
4	5.844	25.053	76%
2	4.571	23.891	80%
1	4.482	21.503	79%

C. *Montage Image Mosaic Workflow*

In the OpenNebula environment, We demonstrate and analyze the integration implementation using a Montage Image Mosaic Workflow. Montage is a suite of software tools developed to generate large astronomical image mosaics by composing multiple small images, as shown in Fig. 4. The typical workflow process involves the following key steps:

- Image projection:
  - re-project each image into a common coordinate space (mProjectPP)
- Background rectification:
  - Calculate a list of overlapping images (mOverlaps)
  - Perform image difference between each pair of overlapping images (mDiffFit)
  - Fit difference images into a plane (mConcatFit)
  - Background correction (mBackground)
- Image co-addition (mAdd):
  - Optionally divide a region into a grid of sub-regions, and co-add the images in each region into a mosaic
  - Co-add the processed images (or mosaics in sub-regions) into a final mosaic

And finally the mosaic is shrunk (mShrink) and converted into a JPEG image (mJPEG) for display.

To visualize the workflow execution, We developed a Nebula Image Mosaic demo service. In the demo a user can pick one of the nebulae (e.g. the Swan Nebulae) to create the mosaic for it, and the demo service submits a workflow request to the Cloud workflow service, which in turn instantiates the Cloud resources on-the-fly to execute the workflow. The demo also visualizes workflow progress in a DAG (directed acyclic graph) on the top, and displays the execution log on the lower left and intermediate results on the lower right, as illustrated in Fig. 5. The deployment provides scientists with an easy-to-use platform to manage and execute scientific workflows on a Cloud platform without knowing the details of workflow scheduling and Cloud resource provisioning.

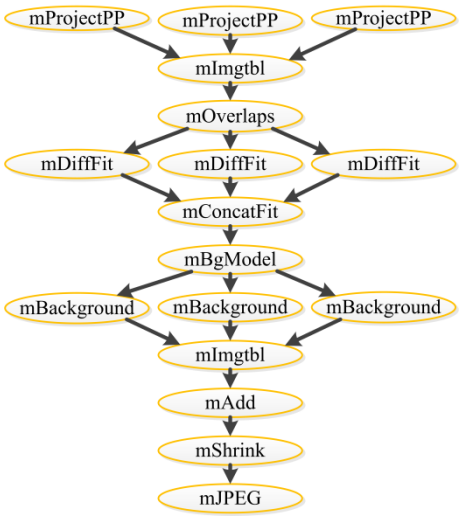

Fig. 4. The Montage Workflow

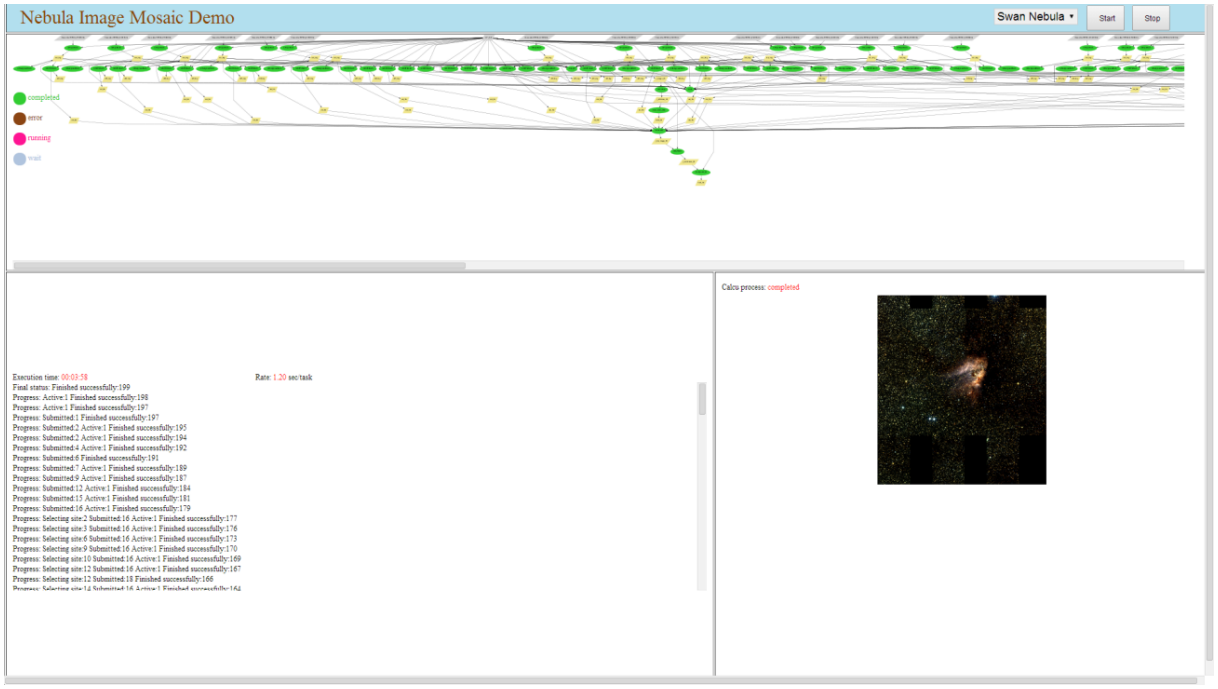

Fig. 5. Nebula Image Mosaic Demo

VI. Conclusions and Future Work

We discuss the challenges for traditional scientific workflow applications in the big data era and the available solutions to deal with the these challenges. We propose a service framework, which meets all the essential requirements for a scientific computing Cloud platform, to normalize the integration of SWFMS and Cloud computing and address the big data processing problem in traditional infrastructures. Meanwhile, we also present our implementation details based on the service framework for the integration of the Swift workflow management system with both OpenNebula and Eucalyptus, and set up a series of experiments to demonstrate the capability of our implementation. We also demonstrate the functionality and efficiency of our approach using a Montage Image Mosaic Workflow.

For future work, we will investigate to port different SWFMSs, such as Taverna, VIEW, etc., to Clouds according to the proposed framework. We will also investigate autonomous application deployment in resource provisioning, which can deploy workflow applications automatically in a virtual cluster.

Acknowledgment

This paper is supported by the key project of National Science Foundation of China No. 61034005 and No. 61272528.

References

[1] Woitaszek, M., Dennis, J., Sines, T. Parallel High-resolution Climate Data Analysis using Swift. 4th Workshop on Many-Task Computing on Grids and Supercomputers 2011.

[2] Damkliang K, Tandayya P, Phusantisampan T, et al. Taverna Workflow and Supporting Service for Single Nucleotide Polymorphisms Analysis[C]//Information Management and Engineering, 2009. ICIME'09. International Conference on. IEEE, 2009: 27-31.

[3] Zhang J, Votava P, Lee T J, et al. Bridging VisTrails Scientific Workflow Management System to High Performance Computing[C]//Services (SERVICES), 203 IEEE Ninth World Congress on. IEEE, 2013: 29-36.

[4] Zhang J. Ontology-driven composition and validation of scientific grid workflows in Kepler: a case study of hyperspectral image processing[C]//Grid and Cooperative Computing Workshops, 2006. GCCW'06. Fifth International Conference on. IEEE, 2006: 282-289.

[5] Juve G, Deelman E. Scientific workflows in the cloud[M]//Grids, Clouds and Virtualization. Springer London, 2011: 71-91.

[6] I. Foster, Y. Zhao, I. Raicu, S. Lu. "Cloud Computing and Grid Computing 360-Degree Compared," IEEE Grid Computing Environments (GCE08) 2008, co-located with IEEE/ACM Supercomputing 2008. Austin, TX. pp. 1-10

[7] G. Bell, T. Hey, A. Szalay, Beyond the Data Deluge, Science, Vol. 323, no. 5919, pp. 1297-1298, 2009.

[8] E. Deelman et al. Pegasus: A framework for mapping complex scientific workflows onto distributed systems, Scientific Programming, vol. 13, iss. 3, pp. 219-237. July 2005.

[9] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, Y. Zhao, Scientific workflow management and the Kepler system, Concurrency and Computation: Practice and Experience,Special Issue: Workflow in Grid Systems, vol. 18, iss. 10, pp. 1039–1065, 25 August 2006.

[10] J. Freire, C. T. Silva, S. P. Callahan, E. Santos, C. E. Scheidegger and H. T. Vo, Managing Rapidly-Evolving Scientific Workflows, Provenance and Annotation of Data, Lecture Notes in Computer Science, 2006, vol. 4145/2006, 10-18, DOI: 10.1007/11890850_2

[11] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. Pocock, P. Li, and T. Oinn, "Taverna: a tool for building and running workflows of services," Nucleic Acids Research, vol. 34, pp. 729-732, 2006.

[12] J.-S. Vöckler, G. Juve, E. Deelman, M. Rynge, G. B. Berriman. "Experiences Using Cloud Computing for A Scientific Workflow Application", Invited Paper, ACM Workshop on Scientific Cloud Computing (ScienceCloud) 2011. pp. 15-24.

[13] Lin C, Lu S, Lai Z, et al. Service-oriented architecture for VIEW: a visual scientific workflow management system[C]//Services Computing, 2008. SCC'08. IEEE International Conference on. IEEE, 2008, 1: 335-342.

[14] Kozlovszky M, Karoczkai K, Marton I, et al. Enabling generic distributed computing infrastructure compatibility for workflow management systems[J]. Computer Science, 2012, 13(3): 61-78.

[15] Wang L, Duan R, Li X, et al. An Iterative Optimization Framework for Adaptive Workflow Management in Computational Clouds[C]//Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on. IEEE, 2013: 1049-1056.

[16] Jrad F, Tao J, Streit A. A broker-based framework for multi-cloud workflows[C]//Proceedings of the 2013 international workshop on Multi-cloud applications and federated clouds. ACM, 2013: 61-68.

[17] K. Keahey, T. Freeman, "Science Clouds: Early Experiences in Cloud Computing for Scientific Applications," Cloud Computing and Its Applications 2008 (CCA-08), Chicago, IL. October 2008.

[18] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, J. Good, "On the Use of Cloud Computing for Scientific Workflows," 3rd International Workshop on Scientific Workflows and Business Workflow Standards in e-Science (SWBES), pp. 640-645, 2008.

[19] M. Wilde, I. Foster, K. Iskra, P. Beckman, Z. Zhang, A. Espinosa, M. Hategan, B. Clifford, I. Raicu, "Parallel Scripting for Applications at the Petascale and Beyond," IEEE Computer Nov. 2009 Special Issue on Extreme Scale Computing, vol. 42, iss. 11, pp. 50-60, 2009.

[20] Y. Zhao, M. Hategan, B. Clifford, I. Foster, G. Laszewski, I. Raicu, T. S.-Praun, M. Wilde. "Swift: Fast, Reliable, Loosely Coupled Parallel Computation," IEEE Workshop on Scientific Workflows 2007, pp. 199-206.

[21] M. Christie and S. Marru. The lead portal: a teragrid gateway and application service architecture: Research articles. Concurrency and Computation : Practice and Experience, 19(6):767{781, 2007.

[22] Bhandarkar M. MapReduce programming with apache Hadoop[C]//Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on. IEEE, 2010: 1-1.

[23] Chaisiri S, Bong Z, Lee C, et al. Workflow framework to support data analytics in cloud computing[C]//Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on. IEEE, 2012: 610-613.

[24] Keahey, K., and T. Freeman. Contextualization: Providing One-click Virtual Clusters. in eScience. 2008, pp. 301-308. Indianapolis, IN, 2008.

[25] C. Lin, S. Lu, X. Fei, A. Chebotko, D. Pai, Z. Lai, F. Fotouhi, and J. Hua, "A Reference Architecture for Scientific Workflow Management Systems and the VIEW SOA Solution," IEEE Transactions on Services Computing (TSC), 2(1), pp.79-92, 2009.

[26] G. Juve and E. Deelman. Wrangler: Virtual Cluster Provisioning for the Cloud. In HPDC, pp. 277-278, 2011.

[27] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, M. Wilde. "Falkon: a Fast and Light-weight tasK executiON framework," IEEE/ACM SuperComputing 2007, pp. 1-12.

[28] Lacroix Z, Aziz M. Resource descriptions, ontology, and resource discovery[J]. International Journal of Metadata, Semantics and Ontologies, 2010, 5(3): 194-207.

[29] Szabo C, Sheng Q Z, Kroeger T, et al. Science in the Cloud: Allocation and Execution of Data-Intensive Scientific Workflows[J]. Journal of Grid Computing, 2013: 1-20.