Architecting Cloud Workflow: Theory and Practice

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

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

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

Xuan Zhang
School of Computer Science and Engineering
Univ. of Electronic and Science Technology of China
Chengdu, China
Jossie.bunny@gmail.com

***Abstract*—The data scale, science analysis and processing complexity in scientific community are growing exponentially in the "big data" era. Cloud computing paradigm has been widely adopted to provide unprecedented scalability and resources on demand, while scientific workflow management systems (SWFMSs) have been proven essential to scientific computing and services computing. Uniting the advantages of both cloud computing and SWFMSs can bring a valuable solution to the scientific "big data" problem to researchers. Although a series of work have concentrated on integrating SWFMSs with Cloud platforms that provide much experience for future research and development, a study from an architectural perspective is still missing. The main contributions of this paper are: 1) based on a comprehensive survey of the available integration options, we propose a service framework for integrating SWFMSs with Cloud computing; 2) we implement the service framework based on various Cloud platforms to validate the feasibility of the proposed framework; and 3) we conduct a set of experiments to demonstrate the capability and use a NASA MODIS image processing workflow as a showcase of the implementation.**

Keywords—Cloud Workflow; Service Framework; Workflow-as-a-Service; Swift; OpenNebula; Eucalyptus

I. INTRODUCTION

Industrial and Scientific communities are facing a "data deluge" [7] coming from products, sensors, satellites, experiments and simulations. Scientists, manufacturers and developers are attempting multifarious methods to deal with the ever-increasing computing and storage problems arising in the "big data" era. As an emerging computing paradigm, Cloud computing [6] is gaining tremendous momentum in both academia and industry. Scientific workflow management systems (SWFMSs) have been proven essential to scientific computing and services computing as they provide functionalities such as workflow specification, process coordination, job scheduling and execution, provenance tracking, and fault tolerance.

Uniting the advantages of both cloud computing and SWFMSs can bring a valuable solution to the scientific "big data" problem to researchers. Cloud offers unprecedented scalability to workflow systems, and could potentially change the way we perceive and conduct scientific experiments. The scale and complexity of the science problems that can be handled can be greatly increased on the Cloud, and the on-demand resource allocation on the Cloud will also help improve resource utilization and user experience.

In many cases, large simulations are organized as scientific workflows that run on Distributed Computing Infrastructures (DCIs), and we realize that workflow management systems are diverse in many aspects, such as workflow models, workflow languages, workflow engines, and so on. In many cases, one workflow system engine is dependent on one specific DCI, porting a workflow management system to run on another DCI may cost a large quantity of extra effort. So in practice, researchers may choose to integrate a specific SWFMS into a particular Cloud, whichever takes the minimum effort to migrate. We expect that the availability of such a service framework can provide a solution to breaking the limitations that a specific SWFMS is bound to a particular Cloud environment and a guidance for the architectural design of integrating SWFMSs into Cloud platforms. To address this issue,

- First, we propose a generic service framework to integrate SWFMSs with various Cloud based DCIs, which covers a wide spectrum from workflow management and migration into Clouds, task scheduling, Cloud resource management, and virtual resource provisioning and recycling.
- Second, through the introduction of the reference service framework, we implement the framework based on a set of open-source and implemented systems to validate the feasibility of the proposed framework.
- Third, we conduct a series of experiments to demonstrate the capability and use a NASA MODIS image processing workflow as a showcase of the implementation.

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

II. Related Work

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 [5] [19] presented a broker-based framework to support the execution of workflow applications on a multi-Cloud environment. Bhaskar Prasad Rimal et al. [4] discussed a framework of scientific workflow for multi-tenant cloud orchestration environment that deals with semantic-based workflow as well as policy-based workflow. To isolate each tenant, they designed three layers of metadata, including tenant-specific metadata, common metadata and data, and maintained them in the metadata repositories which were shared between tenants.

The CODA framework [3] 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, has been integrated into the framework. Through the CODA framework, the workflows can be easily composed and efficiently executed in Amazon EC2. Sunflower [13] was an adaptive P2P agent-based framework for configuring, enacting, managing and adapting autonomic workflows on hybrid Grid-Cloud infrastructures. To orchestrate Grid and Cloud services, Sunflower utilized a bio-inspired autonomic choreography model and integrated the scheduling algorithm with a provisioning component that can dynamically launch virtual machines in a Cloud infrastructure to provide on-demand services in peak-load situations.

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.

A workflow-oriented cloud computing framework, called WfOC [14], was introduced to support workflow-oriented application on multiple data centers. This framework included workflow-oriented cloud computing programming language, tasks extraction and composition, tasks and data sources registration, tasks functions mapper/reducer and other components, and enabled users to especially focus on workflow definition and workflow tasks logic implementation without needing to worry about the distribution of data and target execution systems.

Xiao Liu et al. [16] proposed a generic QoS framework covering the major stages of a workflow lifecycle, for cloud workflow systems. The framework consisted of four components: 1) QoS requirement specification, 2) QoS-aware service selection, 3) QoS consistency monitoring 4) and QoS violation handling. They also illustrated a concrete performance framework as a case study and evaluated the effectiveness of the performance framework in their cloud workflow system.

Those works mentioned above were mainly focused on 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, Researching and designing of a service-oriented framework can help to standardize the integration procedure and interaction between essential systems.

III. Service Framework

In this section, we first present the available options for running scientific workflow within Cloud environment based on different layers of SWFMSs. Then we discuss the service framework and analyze the details from different aspects, including layers, subsystems and interfaces.

A. Available Options

The reference architecture for SWFMSs [22] is proposed as an endeavor to standardize the SWFMS research and development efforts. As shown in Fig. 1, the reference architecture consists of 4 logical layers, 7 major functional subsystems, and 6 interfaces. The first layer is the Operational Layer, which consists of a wide range of heterogeneous and distributed data sources, software tools, services, and their operational environments, including high-end computing environments. The second layer is called the Task Management Layer. This layer consists of three subsystems: Data Product Management, Provenance Management, and Task Management. The third layer, called the Workflow Management Layer, consists of Workflow Engine and Workflow Monitoring. Finally, the fourth layer – the Presentation Layer, consists of the Workflow Design subsystem and the Presentation and Visualization subsystem. The reference architecture would allow the scientific workflow community to focus on different layers and subsystems of SWFMSs, and also enable such systems to interact and interoperate with each other based on the interface definitions.

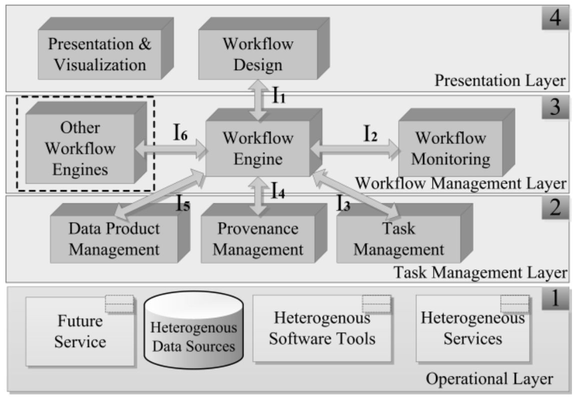


Fig. 1. A reference architecture for SWFMSs

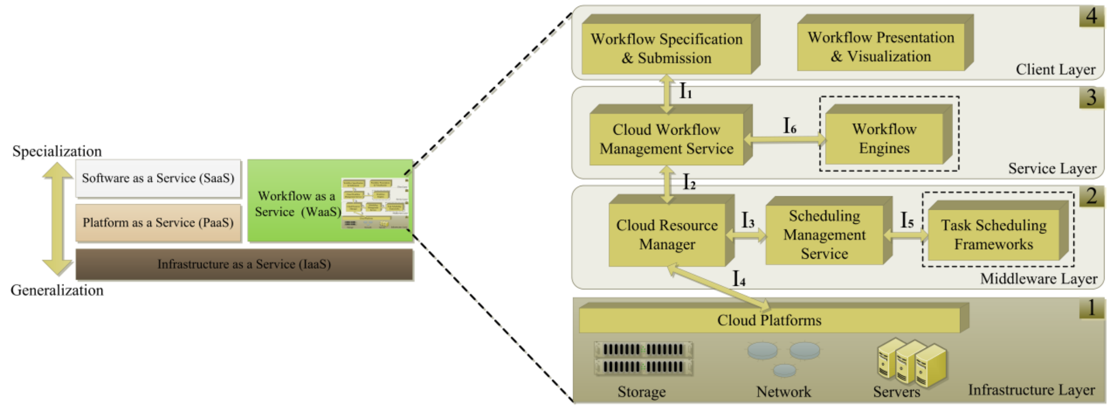
We argue that the above reference architecture is still valid for a Cloud-enabled SWFMS. Here, we consider four possible solutions for deploying the proposed reference architecture in a Cloud computing environment:

1) Operational-Layer-in-the-Cloud. In this solution, only the Operational Layer lies in the Cloud with an SWFMS running out of the Cloud. An SWFMS can now leverage Cloud applications as another type of task components. In contrast to other applications, Cloud-based applications can take advantage of the high scalability provided by the Cloud and the infinite resource capacity provisioned by large data centers. This solution also relieves a user the concern of vendor lock-in due to the relative ease of using alternative Cloud platforms for running Cloud applications. However, the SWFMS itself cannot benefit from the scalability offered by the Cloud.

2) Task-Management-Layer-in-the-Cloud. In this solution, both the Operational Layer and the Task Management Layer will be deployed in the Cloud. In contrast to traditional deployment strategies, Data Product Management, Provenance Management, and Task Management can now leverage the high scalability provided by the Cloud. In particular, Data Product Management and Provenance Management can take advantage of the data models provided by the Cloud, such as blobs, tables, and queues provided by Microsoft Azure. In the meanwhile, Task Management, rather than accommodating the user's request based on a batch-based scheduling system, all-ready tasks can now be immediately deployed over some Cloud computing nodes and get executed instead of waiting in a job queue for the availability of resources. One limitation of this solution is that the economic cost associated with the storage of provenance and data products in the Cloud. Possible workflow tasks might also be restricted to the types of applications and environments (VM instances created by images) that are supported by a particular Cloud infrastructure, which is yet to be standardized. Moreover, although task scheduling and management can benefit from the scalability offered by the Cloud, workflow scheduling and management are not since the workflow engine runs outside of the Cloud.

3) Workflow-Management-Layer-in-the-Cloud. In this solution, the Operational Layer, the Task Management Layer, and the Workflow Management Layer are deployed in the Cloud with the Presentation Layer deployed at a client machine. This solution provides a good balance between system performance and usability: the management of computation, data, and storage and other resources are all encapsulated in the Cloud, while the Presentation Layer remains at the Client machine to support the key architectural requirement of user interface customizability and user interaction support [8]. Such a solution is also most suitable for a scientific workflow application system in which ad hoc domain-specific requirements are constantly evolving, demanding constant changes to the Presentation Layer for that domain. In this solution, both workflow and task management can benefit from the scalability offered by the Cloud, but the downside is that they become more dependent on the Cloud platform over which they run.

4) All-in-the-Cloud. In this solution, a whole SWFMS is deployed inside the Cloud and accessible via a Web browser. A distinct feature of this solution is that no software installation is needed for a scientist to use an SWFMS and an SWFMS can fully take advantage of all the services provided in a Cloud infrastructure. Moreover, the Cloud-based SWFMSs can provide highly scalable scientific workflow 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. One way to address such a concern is to use an on-premise Cloud or a hybrid Cloud, in which public Clouds are used only for shifting out peak workloads.


Specialization
Software as a Service (SaaS)
Platform as a Service (PaaS)
Infrastructure as a Service (IaaS)
Workflow as a Service (WaaS)
Generalization
Workflow Specification & Submission
Workflow Presentation & Visualization
Client Layer
4
I1
Cloud Workflow Management Service
I6
Workflow Engines
Service Layer
3
I2
Cloud Resource Manager
I3
Scheduling Management Service
I5
Task Scheduling Frameworks
Middleware Layer
2
I4
Cloud Platforms
1
Storage
Network
Servers
Infrastructure Layer

(a) (b)

Fig. 2. The Service Framework

As we described, each of the above solutions has its cons and pros. In practice, a hybrid approach might be desirable, in which for each layer, one subsystem or a piece of the subsystem is deployed in the Cloud, while the rest is deployed outside of the Cloud. For each solution, a refined microarchitecture for each layer and subsystem is an important research problem. We envision that in the future, many solution instances of the proposed reference architecture will coexist, each optimized for a particular deployment strategy. In the meanwhile, as each solution instance conforms to the same deployment-strategy-independent reference architecture, interoperability is ensured.

B. Service Framework

For easy integration with a Cloud platform, a *"Task-Management-layer-in-the-Cloud"* approach can be chosen by implementing, for instance an "Amazon EC2" provider to Swift, then tasks in a Swift workflow can be submitted into

EC2 and executed on EC2 VM instances. However, this approach would leave most of the workflow management and dynamic resource scaling outside the Cloud. For application developers, we would like to free them from complicated Cloud resource configuration and provisioning issues, and provide them with the convenience and transparency to scalable Cloud resources, therefore we choose to take the *"Workflow-Management-Layer-in-the-Cloud"* approach, which requires minimal configuration at the client side and supports easy deployment with virtualization techniques.

We propose a structured service framework that covers all the major aspects involved in the migration and integration of SWFMSs 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. 2b, the service framework includes 4 layers, 8 components and 6 interfaces. Fig. 2a shows a typical service stack of Cloud computing: on top of the IaaS layer, the WaaS is designed to provide workflow as a service for researchers and application developers. We position the WaaS layer across both the Saas and PaaS layer, because our proposed service framework can also be applied to provide workflow platform as a service for related scientists.

C. Layers

The first layer is the ***Infrastructure*** Layer, which consists of multiple Cloud platforms with the underlying server, storage and network resources. This layer provides IaaS level support such as the management of the fundamental physical equipment, virtual machines and storage systems to upper layers. The separation of the Infrastructure Layer from other layers isolates the science-focused and technology-independent problem solving environment from the underlying fast advancing high-end computing infrastructure.

The second layer is called the ***Middleware*** Layer. This layer is responsible for resource management and provisioning, and responding to requests from upper-layer and supporting various scheduling frameworks. All the operations that need to access the underlying resources are encapsulated in this layer. According to the description in the Integration Options section, this layer is responsible for the requirements requested by the *Task-Management-Layer-in-the-Cloud* option. Moreover, the separation of the Middleware Layer from the Infrastructure Layer promotes the extensibility of the Infrastructure Layer with new Cloud platforms and new high-end computing facilities, and localizes system evolution due to hardware or software advances to the interface between the Infrastructure Layer and the Middleware Layer.

The third layer is the ***Service*** Layer, which is responsible for providing scientific workflow management as a service to the upper clients and realizing the execution and monitoring of scientific workflows. This layer also provides interfaces to support various workflow engines. According to the integration options, the Service Layer fulfills the requirements addressed in the *Workflow-Management-Layer-in-the-Cloud* option. The separation of the Service Layer from the Middleware Layer concerns two aspects: 1) it isolates the choice of a workflow model from the choice of a task model, so changes to the workflow structure do not need to affect the structures of tasks and 2) it separates workflow scheduling from task execution, thus provides space for performance and scalability of the whole management system.

The fourth layer is the ***Client*** Layer, which provides the functionality of workflow design, specification, visualization and various user interfaces and tools for workflow submission, resource configuration etc. The Client layer may be out of the Cloud to circumvent the disadvantages discussed in the *All-in-the-Cloud* option. The separation of the Client Layer from other layers provides the flexibility of customizing the user interfaces of the system and promotes the reusability of the rest of system components for different scientific domains.

D. Subsystems

The eight major functional subsystems correspond to the key functionalities required for workflow management as a service in the Cloud. Although the reference framework may allow the introduction of additional subsystems and their features in each layer, this paper only focuses on the major subsystems and their essential functionalities.

The *Workflow Specification & Submission* subsystem is responsible for producing workflow specifications represented in a workflow specification language that supports a particular workflow model, and the submission of workflows to the Cloud Workflow Management Service subsystem. The Workflow Specification & Submission subsystem may provide users with a standalone or Web-based workflow designer, which may support both graphical- and scripting-based design interfaces, and a workflow submission component to submit workflows. The interoperability of workflows should be addressed in this subsystem by the standardization and conversion of workflow languages.

The *Workflow Presentation & Visualization* subsystem is important especially for data-intensive and visualization-intensive scientific workflows, in which the presentation of workflows and visualization of various data products and provenance metadata in multi-dimensions are key to gaining insights and knowledge from large amount of data and metadata.

The *Cloud Workflow Management Service* subsystem acts as an intermediary between the workflow client and the backend Cloud Resource Manager, and is the key service in the service framework provided to researchers interested in using Cloud-based scientific workflow. It supports the following functionalities: workflow language compilation, workflow scheduling, resource acquisition, and status monitoring. In addition, the implementation of fault-tolerance mechanism can also be defined in the service.

The *Workflow Engines* subsystem supports various workflow engines and can be specified by end-users from the Workflow Specification & Submission subsystem. A workflow engine is the heart of a workflow system and responsible for creating and executing workflow runs according to a workflow run model, which defines the state transitions of each scientific workflow and its constituent task runs. A workflow run consists of a coordinated execution of tasks, each of which is called a task run. The interoperability of workflows should be addressed by the standardization of interfaces, workflow models, and workflow run models, so that a scientific workflow or its constituent sub-workflows can be scheduled and executed in multiple Workflow Engines that are provided by various vendors.

The *Cloud Resource Manager (CRM)* subsystem is a resource management framework that bridges Cloud Workflow Management Service with various Cloud platforms. It provides scientific workflows with Cloud resource provisioning as a service and the workflows can benefit from the scalability offered by the Cloud. Meanwhile, the dependency on Cloud platforms can be reduced as implementations for various Cloud platforms can be provided, ranging from commercial to open source ones, including Amazon EC2, OpenNebula, Eucalyptus, CloudStack, etc.

The *Scheduling Management Service* subsystem is a framework that bridges Cloud Resource Manager with various Task Scheduling Frameworks. It provides a set of operations for the deployment and management of various scheduling frameworks according to configurations specified by users.

The *Task Scheduling Frameworks* subsystem consists of multiple scheduling frameworks, such as Falkon[20], Sparrow, Gearman, and so on, and the framework can be specified by end-users through configuration. It is devised to schedule tasks delivered from the Workflow Engines subsystem.

The *Cloud Platforms* Subsystem refers to various supported Cloud platforms in general and the functionalities can be summarized from the Infrastructure Layer.

E. Interfaces

In the reference framework, six interfaces are explicitly defined, which show how each subsystem interacts with other subsystems. The interoperability between the subsystems should be addressed by standardizing the interfaces provided by each subsystem.

Interface I_1 provides a set of interfaces for the communication between Workflow Specification & Submission subsystem and the Cloud Workflow Management Service, so workflow specifications created by workflow design tools can be submitted to a workflow execution environment for compiling, scheduling, and management. Interface I_2 provides a series of interfaces for Cloud Workflow Management Service to interact with Cloud Resource Manager: the Cloud Workflow Management Service sends resource request to allocate specified cluster resources, and the Cloud Resource Manager replies with the cluster information for task execution. Interface I_3 provides a series of interfaces for the Cloud Resource Manager to communicate with the Scheduling Management Service: upon the specified resource requests from Cloud Workflow Management Service are received, the Cloud Resource Manager provisions resources and deploys the user-specified Task Scheduling Framework into the cluster based on the services provided by the Scheduling Management Service, then sends cluster information back to the Cloud Workflow Management Service. Interface I_4 provides a set of interfaces for the Cloud Resource Manager to interact with underlying Cloud Platforms, mostly for resource provisioning, monitoring and recycling. Interface I_5 provides a series of interfaces for the Scheduling Management Service to interact with Task Scheduling Frameworks subsystem: the supported operations upon scheduling frameworks are defined here. Interface I_6 provides a set of interfaces to interoperate with deployed Workflow Engines. Workflow Specifications can be passed through to default or user-specified workflow engine for execution.

F. Discussion

The motivation of our work is to break through workflows' dependence on the underlying resource environment, and take advantage of the scalability and on-demand resource allocation of the Cloud. We present a layered service framework for the implementation and application of integrating SWFMSs 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.

IV. Implemntation and Experiment

In this section, we first describe our experience in integrating the Swift scientific workflow management system [10] with different Cloud platforms based on the service framework. Then we show our experiment results of implementation for both the OpenNebula [1] and Eucalyptus [9] platforms to demonstrate the practicability and capability of the service framework.

A. Implementation Architecture & Interfaces

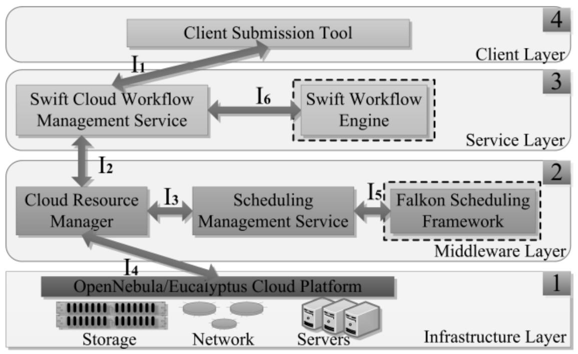


Fig. 3. Integration Architecture

We implement the service framework for both the OpenNebula and Eucalyptus platforms and we show the integration architecture in Fig. 3. The implementation 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 proposed service framework.

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. 2, in Table 1. And we also point out which subsystems are directly from the original systems and which are implemented for the integration.

We also define a series of interfaces to standardize the complicated interactions between different essential subsystems. We list the key interfaces in Table 2, and point out the implementation status and interaction relationships. Further details about these interfaces are available at our website[1].

TABLE I. SUBSYSTEMS IMPLEMENTATION DESCRIPTION

Components	Description	Subsystems
OpenNebula / Eucalyptus	reuse	Cloud Platforms ***(Abbr. CP)***
Falkon Scheduling Framework	minor revision	Task Scheduling Frameworks ***(Abbr. TSF)***
SMS	implemented	Scheduling Management Service ***(Abbr. SMS)***
CMR	implemented	Cloud Resource Manager ***(Abbr. CRM)***
Swift System	minor revision	Workflow Engines ***(Abbr. WE)***
CWMS	implemented	Cloud Workflow Management Service ***(Abbr. CWMS)***
Client Submission Tool	implemented	Workflow Specification & Submission***(Abbr. WSS)***

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.

TABLE II. INTERFACES IMPLEMENTATION DESCRIPTION

Interfaces	Description	Interaction Between
Interface I_1	implemented	***WSS*** and ***CWMS***
Interface I_2	implemented	***CWMS*** and ***CRM***
Interface I_3	implemented	***CRM*** and ***SMS***
Interface I_4	implemented	***CRM*** and ***CP***
Interface I_5	under evaluation	***SMS*** and ***TSF***
Interface I_6	under evaluation	***CWMS*** and ***WE***

a. "implemented": we define and implement the interfaces.

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

B. *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 [17], 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, as illustrated in Fig. 4. 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 cluster, a multi-site Grid, or the Amazon EC2 service.

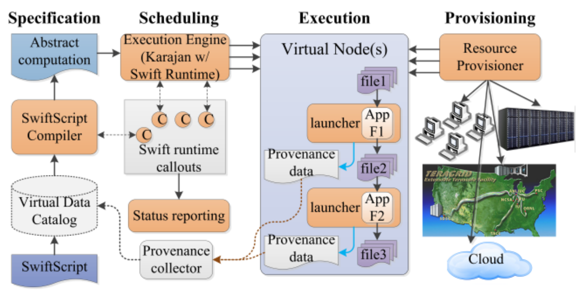


Fig. 4. Swift System Architecture

The four major components of the Swift system can be easily mapped into the four layers in the SWFMSs reference architecture: the specification falls into the Presentation Layer, although SwiftScript focuses more on the parallel scripting aspect for user interaction than on Graphical representation; the scheduling components correspond to the Workflow Management Layer; the execution components maps to the Task Management layer; and the provisioning layer can be thought as mostly in the Operational Layer.

C. *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.

Eucalyptus: Considering the efficient and convenient service provided by the FutureGrid[2], 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. The instance type used in our experiment is m1.small: 1 CPU Unit, 1 CPU Core and 500MB Memory. And the operating system is Ubuntu Server 12.04.

D. *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

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

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

results be more intuitionistic and comparable. We measure the performance to establish a baseline for resource provisioning and Cloud resource management overhead in the science Cloud environment.

1) The base line measurement

We first measure the base line for server initialization time and worker registration time. We request a Falkon virtual cluster with 1 server, and varying number of workers.

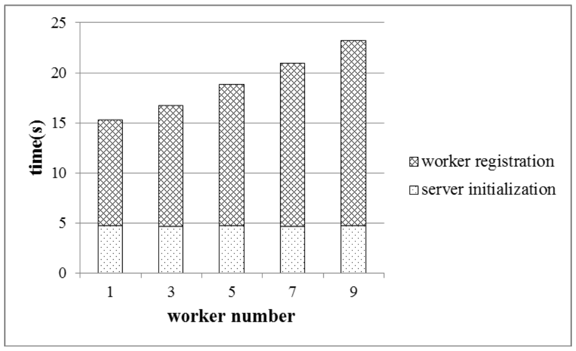


Fig. 5. The Base Line Measurement (OpenNebula)

The base line results in Fig. 5 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.

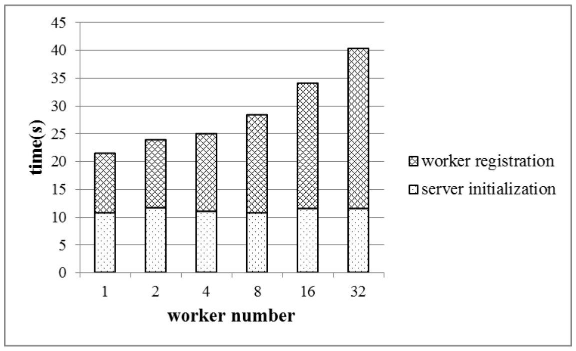


Fig. 6. The Base Line Measurement (Eucalyptus)

We measure the server initialization time and worker registration time (illustrated in Fig. 6) 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. 5. 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. 5.

2) The resource recycling measurement

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.

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 (as shown in Fig. 7).

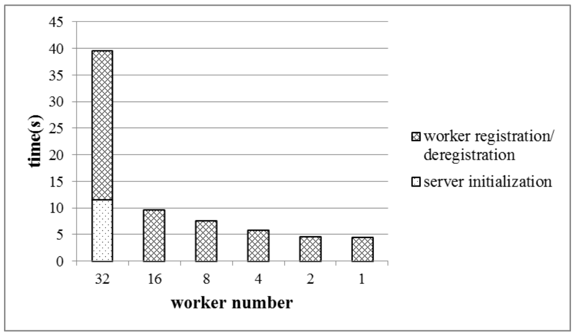


Fig. 7. Decreasing Resource Required (Eucalyptus)

E. MODIS Image Processing Workflow

We demonstrate and analyze the integration implementation in OpenNebula environment using a NASA MODIS [11] image processing workflow. The NASA MODIS dataset we use is a set of satellite aerial data blocks, each block is of size around 5.5MB, with digits indicating the geological feature of each point in that block, such as water, sand, green land, urban area, etc.

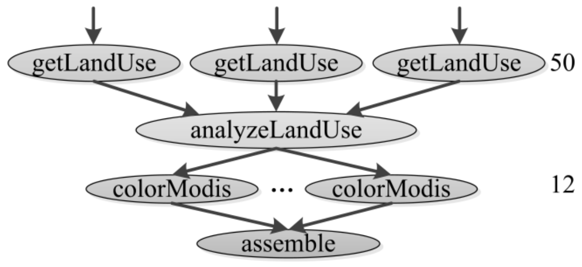


Fig. 8. MODIS Image Processing Workflow

The workflow (illustrated in Fig. 8) takes a set of such blocks, gets the size of the urban area in each of the blocks, analyzes and picks the top 12 of the blocks that have the largest urban area, converts them into displayable format, and assembles them into a single PNG file.

In this experiment, we change the number of input data blocks from 50 blocks to 25 blocks, and measure the total execution time with varying number of workers in the virtual cluster. In Fig. 9, we can observe that with the increase of the number of workers, the execution time decreases accordingly (i.e. execution efficiency improves), however at 5 workers to process the workflow, the system reaches efficiency peak. After that, the execution time goes up with more workers. This means that the improvement cannot subsidize the management and registration overhead of the added worker. The time for server initialization and worker registration remain unchanged when we change the input size (as have been shown in Fig. 5).

The experiment indicates that while our virtual resource provisioning overhead is well controlled, we do need to carefully determine the number of workers used in the virtual cluster to achieve resource utilization efficiency, which will be tuned in our future research endeavor.

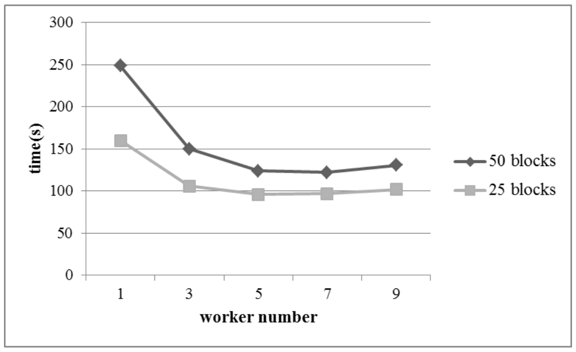


Fig. 9. Different Input Sizes (OpenNebula)

V. Conclusions and Future Work

We propose a reference service framework for migrating SWFMSs into Cloud to take advantage of Cloud scalability, and also to handle the ever increasing data scale and analysis complexity of scientific applications. We present our implementation effort in integrating the Swift workflow management system with the OpenNebula and the Eucalyptus Cloud platforms according to the service framework, in which a client-side tool, a Cloud workflow management service, a Cloud resource manager, and a scheduling management service are developed. We also demonstrate the functionality and efficiency of our approach using a real-world scientific workflow.

The implementation can readily be used for Openstack[2] as it is getting more popularity in scientific research area and commercial applications. We are also investigating the integration of other SWFMSs into these various Clouds and the auto-scaling mechanism, which can adjust the number of workers automatically according to workflow workload.

Acknowlegment

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

References

[1] OpenNebula, [Online]. Available: http://www.OpenNebula.org, 2014

[2] Openstack, [Online]. Available: http://www.openstack.org, 2014

[3] 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.

[4] Rimal B P, El-Refaey M A. A framework of scientific workflow management systems for multi-tenant cloud orchestration environment[C]//Enabling technologies: Infrastructures for collaborative enterprises (wetice), 2010 19th ieee international workshop on. IEEE, 2010: 88-93.

[5] M. Kozlovszky, K. Karóczkai, I. Márton, A. Balasko, A. C. Marosi, and P. Kacsuk, "Enabling Generic Distributed Computing Infrastructure Compatibility for Workflow Management Systems", Computer Science, vol. 13, no. 3, p. 61, 2012.

[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] C. Lin, S. Lu, Z. Lai, A. Chebotko, X. Fei, J. Hua, F. Fotouhi, "Service-Oriented Architecture for VIEW: a Visual Scientific Workflow Management System," In Proc. of the IEEE 2008 International Conference on Services Computing (SCC), pp.335-342, Honolulu, Hawaii, USA, July 2008.

[9] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, D. Zagorodnov. The Eucalyptus Open-Source Cloud-Computing System, 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID '09, pp. 124-131, 2009.

[10] 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.

[11] NASA MODIS dataset, [Online]. Available: http://modis.gsfc.nasa.gov/, 2012.

[12] Y. Zhao, X. Fei, I. Raicu, S. Lu, Opportunities and Challenges in Running Scientific Workflows on the Cloud, IEEE International Conference on Cyber-enabled distributed computing and knowledge discovery (CyberC), pp. 455-462, 2011.

[13] Papuzzo G, Spezzano G. Autonomic management of workflows on hybrid grid-cloud infrastructure[C]//Proceedings of the 7th International Conference on Network and Services Management. International Federation for Information Processing, 2011: 230-233.

[14] Pang J, Cui L, Zheng Y, et al. A workflow-oriented cloud computing framework and programming model for data intensive application[C]//Computer Supported Cooperative Work in Design (CSCWD), 2011 15th International Conference on. IEEE, 2011: 356-361.

[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] Liu X, Yang Y, Yuan D, et al. A generic qos framework for cloud workflow systems[C]//Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on. IEEE, 2011: 713-720.

[17] Y. Zhao, J. Dobson, I. Foster, L. Moreau, M. Wilde, "A Notation and System for Expressing and Executing Cleanly Typed Workflows on Messy Scientific Data," SIGMOD Record, vol. 34, iss. 3, pp. 37-43, September 2005.

[18] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer, D. H. Epema, "Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing," IEEE Transactions on Parallel and Distributed Systems, pp. 931-945, June, 2011.

[19] 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.

[20] 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.

[21] K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon et al. Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud. In CloudCom, IEEE, pp. 159–168, 2010.

[22] 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.