ARTICLE IN PRESS

Future Generation Computer Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Enabling scalable scientific workflow management in the Cloud

Yong Zhao [a], Youfu Li [a,*], Ioan Raicu [b], Shiyong Lu [c], Wenhong Tian [a], Heng Liu [a]

[a] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China

[b] Department of Computer Science, Illinois Institute of Technology, Chicago, IL, USA

[c] Department of Computer Science, Wayne State University, Detroit, MI, USA

HIGHLIGHTS

- We analyze the major challenges of running scientific workflows on the Cloud.
- We propose a reference framework to standardize the integration.
- The implementation experience proves that the framework is feasible and extendible.
- Cluster-recycling mechanism can improve the resource provisioning efficiency.

ARTICLE INFO

Article history:
Received 15 February 2014
Received in revised form
14 July 2014
Accepted 23 October 2014
Available online xxxx

Keywords:
Cloud workflow
Virtual resource provisioning
Cloud resource management
Scientific computing platform

ABSTRACT

Cloud computing is gaining tremendous momentum in both academia and industry. In this context, we define the term "Cloud Workflow" as the specification, execution and provenance tracking of large-scale scientific workflows, as well as the management of data and computing resources to support the execution of large-scale scientific workflows in the Cloud. In this paper, we first analyze the gap between these two complementary technologies, and what it means to bring Clouds and workflows together. Then, we present the key challenges in supporting Cloud workflows, and present our reference framework for scientific workflow management in the Cloud. Last we present our experience in integrating a scientific workflow management system—Swift into the Cloud. We discuss the performance of cluster provisioning within the OpenNebula Cloud platform, the Eucalyptus Cloud platform and Amazon EC2, and we demonstrate the capability and efficiency of the integration using a NASA MODIS image processing workflow and the Montage image mosaic workflow.

Note to practitioners. Scientific workflow management plays a very important role for scientific computing and application coordination, while Cloud computing offers scalability and resource on-demand. We devise autonomous methods to integrate scientific workflow management systems with Cloud platforms and also provision resources for large scale workflows, which can facilitate scientists to easily manage their workflows in the Cloud, and take advantage of large scale Cloud resources. There are a few integration options and many challenges in the process, and the experience we gain will help researchers in migrating their workflow management systems and workflow applications into the Cloud.

1. Introduction

Governments, research institutes, and industry leaders are strategically adopting Cloud computing [1], to solve their ever-increasing computing and storage problems arising in the Internet age. There has been a burgeoning of Cloud platforms and applications in both academia and industry: not long after Amazon opened its Elastic Computing Cloud (EC2) to the public, Google, IBM, and Microsoft all released their Cloud platforms one after another. Meanwhile, several open source Cloud platforms, such as Hadoop [2], OpenNebula [3], Eucalyptus [4], Nimbus [5], and OpenStack [6], become available with fast growth of their own communities.

There are a couple of major benefits and advantages that are driving the widespread adoption of the Cloud computing paradigm: (1) Easy access to resources: resources are offered as services and can be accessed over the Internet. (2) Scalability on demand: once an application is deployed onto the Cloud, the application can be automatically made scalable by provisioning the resources in the Cloud on demand, and the capability of scaling out and in, and load

* Corresponding author.
E-mail address: youfuli.fly@gmail.com (Y. Li).

http://dx.doi.org/10.1016/j.future.2014.10.023
0167-739X/

Please cite this article in press as: Y. Zhao, et al., Enabling scalable scientific workflow management in the Cloud, Future Generation Computer Systems (2014), http://dx.doi.org/10.1016/j.future.2014.10.023

balancing; (3) Better resource utilization: Cloud platforms can coordinate resource utilization according to resource demand of the applications hosted in the Cloud; and (4) Cost saving: Cloud users are charged based on their resource usage in the Cloud, they only pay for what they use, and if their applications get optimized, that will be reflected into a lowered cost immediately.

In the meanwhile, scientific workflow has become an increasingly popular paradigm for scientists to formalize and structure complex scientific processes to enable and accelerate many significant scientific discoveries. A scientific workflow management system (SWFMS) is a system that supports the specification, modification, execution, failure handling, and monitoring of scientific workflows using workflow logic to control the order of executing workflow tasks. The importance of scientific workflows has been recognized and re-emphasized in a science article titled "Beyond the Data Deluge" [7], which concluded, "*In the future, the rapidity with which any given discipline advances is likely to depend on how well the community acquires the necessary expertise in database, workflow management, visualization, and cloud computing technologies.*", and Cloud workflow would cover both the workflow and Cloud aspects.

The world has entered into a "big data" era. The amount of data created in the world is growing at an exponential rate. Advances in science instrumentation and network technologies are posing new challenges to our workflow systems in both data scale and application complexity. 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, in addition to data scale and computation complexity, there are also resource provisioning, collaboration in heterogeneous environments, etc. [8]. While Cloud computing provides a high-level abstraction of computing resources as services, scientific workflows provide a high-level abstraction of applications as dataflow-oriented graphs. An emerging trend is to run scientific workflows on the Cloud, thus delivering the benefits of both to end users. Cloud computing provides a paradigm-shifting utility-oriented computing environment in terms of the unprecedented size of datacenter-level resource pool and the on-demand resource provisioning mechanism, enabling scientific workflow solutions that can address peta-scale scientific problems.

In the rest of the paper, we discuss the various aspects involved in running Cloud workflows. In Section 2, we present key challenges in supporting Cloud workflows; In Section 3, we discuss existing research and related work; In Section 4, we present a reference service framework for workflow integration and migration into Cloud and our implementation experience in integrating Swift with OpenNebula and other Cloud platforms; In Section 5, we discuss and compare the results in running Swift within OpenNebula, Eucalyptus and Amazon EC2 commercial Cloud; In Section 6, we point out some research directions for researchers interested in our work, and in Section 7, we draw our conclusions.

2. Challenges

Scientific workflow systems have been formerly applied over a number of execution environments such as workstations, clusters/grids, and supercomputers, where the new Cloud computing paradigm with unprecedented size of datacenter-level resource pool and on-demand resource provisioning can offer much more to such systems, enabling scientific workflow solutions capable of addressing peta-scale scientific problems. The benefit of running scientific workflows on top of 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.
(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 preloaded in virtual machine (VM) images, or deployed dynamically onto VM instances.
(3) The on-demand resource allocation mechanism in 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 nodes upfront. Cloud workflows can scale out and in dynamically, resulting in a fast turn-around time for end users.
(4) Cloud computing provides a 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 a 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.

Despite the advantages and opportunities that Cloud computing can provide for scientific workflows, there are many major obstacles to the adaptation and running of scientific workflows on the Cloud. We have identified some of the key challenges in our prior work [9] and we list them below.

Architectural challenges: Our previously designed scientific workflow management system reference architecture [10] consists of four layers—Operational, Task Management, Workflow Management, and Presentation Layers. To engineer an SWFMS onto Clouds, it may not be as simple as to replace the Operational Layer with a Cloud infrastructure. We need to take a bottom-up approach and evaluate the requirements, look at integration problems at the other three layers as well, to address compatibility and impedance problems that are introduced by different Cloud providers and heterogeneous implementations.

Integration challenges: Many of the immediate challenges to running scientific workflows on the Cloud are to integrate scientific workflow systems with Cloud infrastructure and resources. In most cases, we will need to change the way how an SWFMS acquires resources, dispatches tasks, monitors the progress of those tasks, tracks provenance information [11], and how it deals with errors and exceptions in the Cloud.

Computing challenges: For scientific workflows, leveraging large scale computing resources in the Cloud is not as straightforward as requesting a certain number of computing nodes, there are challenges with regard to resource requirements and provisioning, virtualization, fault tolerance and smart reruns. There are also overheads introduced by some of the technologies employed by most Clouds (e.g. virtualization) which might not be widely accepted by the scientific computing community.

Security challenges: Security has been identified as one of the main concerns for the adoption and success of the Cloud [1] and is the first major service that needs to be provided by a Cloud provider. For example, Microsoft Azure Cloud Platform offers access control as a primary service of the NET Services. To ensure the security for Cloud-based SWFMSs, we can utilize the following three mechanisms: access control, information flow control and secure electronic transaction protocol, depending on scenarios.

Please cite this article in press as: Y. Zhao, et al., Enabling scalable scientific workflow management in the Cloud, Future Generation Computer Systems (2014), http://dx.doi.org/10.1016/j.future.2014.10.023

Data management challenges: Running workflows in the Cloud has to deal with data moving in to and out of the Cloud, large scale data storage within the Cloud, and the exploration of data locality, data and computation co-location issues for efficiency purposes, and the track of data provenance in order to understand and reuse workflows. Furthermore, many scientific workflow systems have been accustomed to global parallel file systems (e.g. GPFS, PFVS, Lustre) as a tool to manage, coordinate, and share data (which is a reasonable assumption in a HPC environment), however most available Clouds do not have such parallel file systems, and mostly rely on distributed object stores (e.g. Amazon S3) which many times break backwards compatibility of the applications (due to a lack of POSIX support) requiring code modification.

Language challenges: Up to now, MapReduce has been the only widely adopted computing model for Clouds, and there are a number of variations of languages based on this model for task specification. Examples are Sawzall [12], Pig [13], and DryadLINQ [14]. However, a workflow specification requires far more functionality and flexibility than what MapReduce can provide, and the implicit semantics incurred by a workflow specification goes far more than just the "map" and "reduce" functional operations. For instance, the mapping of computation to compute node and data partitions, runtime optimization, retry on error, smart re-run, all need to be optimized for the Cloud. The specification and the corresponding implementation of the specification would carry about all the computing and data management challenges associated with interpreting and executing the specification.

Last but not the least, are *service management challenges*, as Clouds are mostly built on top of service oriented architecture, and SWFMSs are also shifting from conventional applications to service systems. We need to deal with service discovery, large input and output handling, data services, and all the other challenges that we are facing in migrating applications into a service world.

We elaborate on architectural challenges, integration challenges, computing challenges and security challenges in the following sections since they are closely related to SWFMS migration into the Cloud.

2.1. Architectural challenges

The reference architecture for SWFMSs [10] is proposed as an endeavor to standardize 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 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.

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 separation of the Operational Layer from other layers isolates data sources, software tools, services, and their associated high-end computing environments from the scope of an SWFMS.

The second layer is called the Task Management Layer. This layer consists of three subsystems: Data Product Management, Provenance Management, and Task Management. The separation of the Task Management Layer from the Operational Layer promotes the extensibility of the Operational Layer with new services and new high-end computing facilities, and localizes system evolution due to hardware or software advances to the interface between the Operational Layer and the Task Management Layer.

The third layer, called the Workflow Management Layer, consists of Workflow Engine and Workflow Monitoring. The separation of the Workflow Management Layer from the Task Management Layer concerns two aspects as follows: (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 improves the performance and scalability of the whole system.

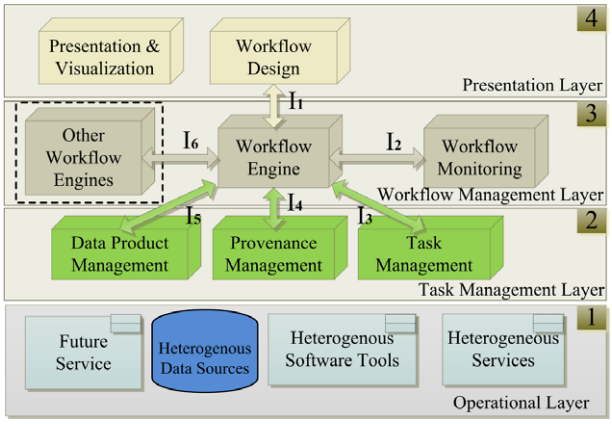


Fig. 1. A reference architecture for SWFMSs.

Finally, the fourth layer—the Presentation Layer, consists of the Workflow Design subsystem and the Presentation and Visualization subsystem. The separation of the Presentation 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.

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 datacenters. 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.

Please cite this article in press as: Y. Zhao, et al., Enabling scalable scientific workflow management in the Cloud, Future Generation Computer Systems (2014), http://dx.doi.org/10.1016/j.future.2014.10.023

(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 [15]. 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 SWFMS 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.

As we described, each of the above solutions has its pros and cons. 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.

2.2. Integration challenges

Many of the immediate challenges to running scientific workflows on the Cloud are to integrate scientific workflow systems with Cloud infrastructure and resources. As we have discussed in the previous section, the degree of integration also depends on how we choose to deploy an SWFMS into Clouds. We identify some of the practical ones below, and also discuss possible solutions to them.

Applications, services, tools integration: In the Operational-Layer-in-the-Cloud approach, we treat applications, services, and tools hosted in the Cloud as task components in a workflow, the scheduling and management of a workflow are mostly outside the Cloud, where these task components are invoked as they are scheduled to execute. The invocation would involve just figuring out the right interface to interact with such applications, services, tools, for instance, via HTTP or REST protocols, or Web services calls, and then in the workflow, it needs to transform the output of one invocation, and feed it as an input to another invocation. A majority of the mashup sites (such as those that leverage Google's map service) take this approach, and they use *ad hoc* scripts and programs or shimming techniques [16,17] to glue the services together. An early exploration of the Taverna workflow engine and gRAVI services in the caBIG project [18] can also be thought as an example of integrating an off-the-shelf workflow engine with Cloud/Grid services. gRAVI can rapidly wrap and expose applications, scripts and workflows as Web services, and deploy them into Grids, or the Nimbus Cloud environment.

We notice that while the approach works for applications with small data exchanges, moving large datasets in and out the Cloud would incur serious overhead. For data intensive applications, it is necessary to migrate data into the Cloud. Loading data in and out the Cloud is not a trivial process, for instance, within Microsoft, to load each day's Bing search log into the Cloud, this task itself takes hundreds of dedicated servers working around the clock. So in an ideal Cloud workflow solution, we should avoid such operations as much as possible.

Once we decide to get task dispatching and scheduling into the Cloud, resource provisioning becomes the next issue to resolve. Although conceptually Cloud offers uncapped resources, and a workflow can request as many resources as it requires, this comes with a cost and the presumption that the workflow engine can talk directly with the resource allocated in the Cloud (which is usually not true without tweaking the configuration of the workflow engine). Taking these two factors into consideration, some existing solutions such as Nimbus would acquire a certain number of virtual machines, and assemble them as a virtual cluster, onto which existing cluster management systems such as PBS can be deployed and used as a job submission/execution service that a workflow engine can directly interact with. Some existing study [19] simply choose manual deployment over automated provisioning, in which the provisioning step involves construction of a virtual Condor pool, where the VMs act as Condor worker nodes and report to a Condor Master node that runs on a submit host outside the Cloud.

Debugging, monitoring, and provenance tracking for workflows become increasingly difficult in a Cloud environment, since compute resources are usually dynamically assigned and based on virtual machine images, the environment that a task is executed on could be destroyed right after the task is finished, and assigned to a complete different user and task. Some Clouds also support task migration where tasks can be migrated to another virtual machine if there is problem with the node that the task was initially running on.

Porting an SWFMS into the Cloud can also be a concern, which would usually involve wrapping up an SWFMS as a Cloud service. To fully explore the capability and scalability of the Cloud, a workflow engine may need to be re-engineered to be able to interact directly with various Cloud services such as storage, resource allocation, task scheduling, and monitoring. At the client side, either a complete Web-based user interface needs to be developed to allow users to specify and interact with the SWFMS (e.g., the latest VIEW [16]), or a thin desktop client application needs to be developed to interact with the SWFMS Cloud service.

2.3. Computing challenges

Although Clouds can potentially offer unlimited resources to an SWFMS, managing large-scale computing resources is not a trivial task. Workflow systems may not be able to talk to Cloud resources directly, and they may need to go through middleware services such as Nimbus [5] and Falkon [20] that handle resource provisioning and task dispatching. Things can be even more complicated if we take into consideration of issues such as workflow resource requirement, data dependencies, Cloud virtualization, etc.

Resource provisioning is the first thing to consider when managing task executions in a Cloud. Different stages of a workflow may require different types of resources, and Cloud virtualization can configure Virtual Machines (VMs) differently to meet such requirements, but to what extent (i.e. at what level of granularity) and how flexible it can be would be hard to decide. Amazon only

Please cite this article in press as: Y. Zhao, et al., Enabling scalable scientific workflow management in the Cloud, Future Generation Computer Systems (2014), http://dx.doi.org/10.1016/j.future.2014.10.023

offers a few EC2 instance types which are coarsely categorized as small, medium and large, and they are charged differently according to the computing power they provide. Resource provisioning also needs to be dynamic since the number of resources required for different stages of a workflow can be different, and having a fixed number of allocation can be either insufficient, or wasteful.

SWFMSs also place special emphasis on fault tolerance and smart reruns. A workflow may involve a large number of computations and the whole process can be lengthy, so typically an SWFMS will try to automatically recover when non-fatal errors happen (by using mechanisms such as retry on error, re-schedule computation to a different resource, etc.). Also, in the case that a workflow has to be stopped, detailed execution information will be logged, and the next time the workflow is re-started, it will be able to pick up from where it is stopped. This is called smart-rerun. In a Cloud environment, the scale of a workflow can be much larger, and more components (such as VMs) can be involved, some extra measures need to be taken to support such features. Task duplication has been adopted, e.g. in Google's MapReduce system, and in Dryad/Scope to handle a problem called "straggler" where out of a large number of tasks, one or a few outliers may run for unusually long time, or eventually fail due to various issues. Although this has resource utilization implications, the overall reliability and turnaround time can be improved dramatically using this mechanism.

2.4. Security challenges

Although much research has been done on workflow security, security for Cloud-based SWFMSs is still preliminary, of which we briefly discuss the following three aspects:

Access control. Access control concerns about which principals have the privileges to access which resources [21,22]. In a Cloud-based SWFMS, the resources include Cloud services, SWFMS services and products such as scientific workflows, tasks, provenance, data products, and other artifacts. Due to the dynamic nature (artifacts can be produced constantly) and the large-scale data, metadata, and service sharing nature of the Cloud, access control is a challenging but important research problem.

Information flow control. Information flow control concerns about to whom a piece of information can be passed on. Since a scientific workflow might orchestrate a large number of distributed services, data, and applications, particularly in a large-scale Cloud environment, the mechanism that controls mission-critical information and intellectual property (e.g., secrete parameters used to run a scientific workflow) not being propagated to an unauthorized user is worth looking into.

Secure electronic transaction protocol. Cloud Computing is one kind of utility computing based on the pay-as-you-go pricing model. A secure electronic transaction protocol is to ensure goods atomicity—a user is charged if and only if a service or resource is used by a user and the charge should be no more and no less. To prevent the abuse of Cloud accounts and double or wrong charges by a Cloud provider, further research might be needed to ensure the security of Cloud-based transaction protocols.

2.5. Mapping VIEW and Swift into the reference architecture

As the reference architecture for SWFMSs can provide a guidance for the architectural design of a particular SWFMS in various scientific domains. We use two real-world scientific workflow systems: VIEW and Swift, as a showcase, and explain in detail how their architecture is mapped into the reference architecture.

2.5.1. The VIEW workflow management system

The Visual sciEntific Workflow management system (VIEW) [23] seamlessly integrates the interoperability, extensibility, and reasoning advantages of Semantic Web technology, the querying and storage power of a RDBMS, and the appealing visual features of visualization techniques.

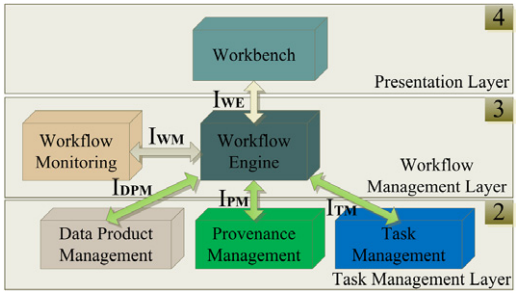


Fig. 2. Overall architecture of the VIEW system.

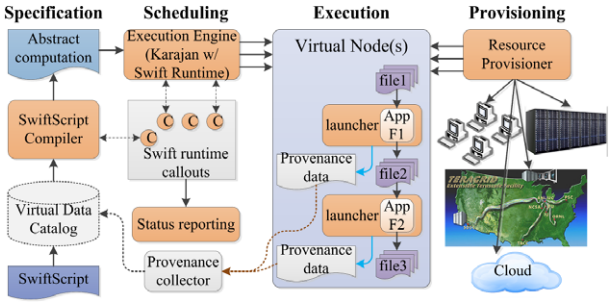


Fig. 3. Swift system architecture.

As shown in Fig. 2, the overall architecture of VIEW [15] is consistent with the reference architecture for SWFMSs. The VIEW system consists of six loosely-coupled, autonomous, reusable, and discoverable service components that correspond to the main functional subsystems proposed in the reference architecture. The Operational Layer is omitted from the figure for simplicity. Except for Workbench, the interface for each service component is defined and described by WSDL: I_{WE}, I_{WM}, I_{TM}, I_{PM} and I_{DPM} for the interface of the Workflow Engine, Workflow Monitor, Task Manager, Provenance Manager, and Data Product Manager, respectively. Service components interact with one another by Web service invocation using SOAP messages via Internet based protocols.

2.5.2. The Swift workflow management system

Swift is a system that bridges scientific workflows with parallel computing. It is a parallel programming system for rapid and reliable specification, execution, and management of large-scale science and engineering workflows. Swift takes a structured approach to workflow specification, scheduling, and execution. It consists of an elegant scripting language called SwiftScript for concise specification of complex parallel computations based on dataset typing and iterations [24], and dynamic dataset mappings for accessing large-scale datasets represented in diverse data formats. The runtime system provides an efficient workflow engine for scheduling and load balancing, and it can interact with various resource management systems such as PBS and Condor for task execution.

The Swift system architecture consists of four major components: Program Specification, Scheduling, Execution, and Provisioning, as illustrated in Fig. 3. 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.

The four major components of the Swift system can be easily mapped into the four layers in the reference architecture: the

Please cite this article in press as: Y. Zhao, et al., Enabling scalable scientific workflow management in the Cloud, Future Generation Computer Systems (2014), http://dx.doi.org/10.1016/j.future.2014.10.023

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.

3. Related work

There have been a couple of early explorers that tried to evaluate the feasibility, performance, and adaptation of running data intensive and HPC applications on Clouds or hybrid Grid/Cloud environments. Palankar et al. [25] evaluated the feasibility, cost, availability and performance of using Amazon's S3 service to provide storage support to data intensive applications, and also identified a set of additional functionalities that storage services targeting data-intensive science applications should support. Oliveira et al. [26] evaluated the performance of X-ray Crystallography workflow using SciCumulus middleware with Amazon EC2. Wang et al. [27] presented their early definition and experience of scientific Cloud computing in the Cumulus project by merging existing Grid infrastructures with new Cloud technologies. These studies provide good sources of information about Cloud platform support for science applications. Other studies investigated the execution of real science applications on commercial Clouds [28,19], mostly being HPC applications, and compared the performance and cost against Grid environments. While such applications indeed can be ported to a Cloud environment, Cloud execution does not show significant benefit due to the applications' tightly coupled nature.

Srirama et al. [29] studied the effects of moving parallel scientific applications onto the Cloud through deploying several benchmark applications (e.g. matrix–vector operations, NAS parallel benchmarks, and DOUG—Domain decomposition On Unstructured Grids) on the Cloud. The authors also observed the limitations of Cloud and its comparison with cluster in terms of performance, and raises important issues around the necessity for better frameworks or optimizations for MapReduce style applications. Parashar et al. [30] proposed a framework to manage autonomics, including scheduling the mix of hybrid resources, application management, monitoring resources and adaptation of resource provisioning based on objectives and metrics. The authors explored the autonomics using a real world scientific workflow, the Defiant reservoir simulator using an Ensemble Kalman Filter with several different objectives (e.g. acceleration, conservation and resilience) and metrics (e.g. deadline and budget). They formed a hybrid infrastructure with TeraGrid and several instance types of Amazon EC2 and the results show that the proposed framework for autonomics works well for the application workflow achieving the objectives using the metrics.

Deelman et al. have studied the cost and performance of workflows in the cloud via simulation [28], using an experimental Nimbus cloud [31], individual Elastic Compute Cloud (EC2) nodes [32], and a variety of different intermediate storage systems on EC2 [33]. Vecchiola et al. have done similar investigations on EC2 and Grid5000 [19]. These studies primarily analyzed the performance and cost of workflows in the cloud, rather than the practical experience of deploying workflows in the cloud. To address the shortages, Juve et al. [34] also related the practical experience of trying to run a nontrivial scientific workflow application on three different infrastructures and compared the benefits and challenges of each platform.

With VGrADS [35], not only the virtual Grid abstraction enabled a more sophisticated and effective scheduling of workflow sets, unifying workflow execution over batch queue systems and cloud computing sites (including Amazon EC2 and Eucalyptus), but also the Virtual Grid Execution System provided a uniform interface for provisioning, querying, and controlling the resources. Its workflow planner could interact with a DAG scheduler, an EC2 planner and fault tolerance sub-components to trade-off various system parameters—performance, reliability and cost.

Table 1
Use cases of SWFMSs.

SWFMSs	Application fields	Use cases
Swift	Climate science	Climate data analysis [41]
Taverna	Bioinformatics	Single nucleotide polymorphisms analysis [42]
Vistrails	Earth science	NASA earth exchange [43]
Kepler	Physics	Hyperspectral image processing [44]
VIEW	Medical science	Neurological disorder diagnosis [15]

SWFMSs such as Taverna [36], Kepler [37], Vistrails [38], Pegasus [39], Swift [40], and VIEW [15] have seen wide adoption in various disciplines such as Physics, Astronomy, Bioinformatics, Neuroscience, Earth Science, and Social Science. In Table 1, we list some use cases that focused on applying SWFMSs to execute data-intensive applications.

Oozie represents a major advancement as a scalable, multi-tenant, secure, and operable workflow service for Hadoop. Islam et al. [45] illustrated the need for Hadoop workflow by describing the requirements not met by existing workflow systems in the large-scale Hadoop computing environments. After discussing the architecture of Oozie, they analyzed the characteristics of the Oozie service in production, followed by some experimentation to quantify the scaling limiters for Oozie. Although Oozie can provide Yahoo and other organizations with major advantages in security, scalability and operability for Hadoop-based applications, simply mapping all the existing mature scientific workflows applications into Oozie-based workflows can be very impracticable and may cost scientists large amount of time beyond their research topics.

Amazon Simple Workflow Service (SWF) defines an interface for workflow orchestration and provides state persistence for workflow executions. Amazon SWF applications involve communication between the following entities: The Amazon Simple Workflow Service, Workflow Executors, Deciders and Activity Workers. However, an important downside of SWF is its incompatibility with existing workflow systems, and lack of means for reusing scientific legacy code [46].

Since both the Oozie and Amazon SWF are bound to particular Cloud platforms, users may need to evaluate the risk associated with vendor lock-in when choosing workflow services. To address these disadvantages, we try to integrate the scientific workflow management systems with various Cloud platforms, without depending on one specific Cloud vendor, to provide a Cloud workflow platform as a service for scientists and developers.

4. Integration framework

In this section we present a general framework that addresses the integration of SWFMSs into the Cloud, and also describe our end-to-end practical application of the framework to integrate Swift [40], an SWFMS that has broad application in Grids and supercomputers [47], with the OpenNebula Cloud computing platform. The integration covers all the major aspects involved in workflow management in the Cloud, from client-side workflow submission to the underlying Cloud resource management, thus enabling scientific workflow management in the Cloud. We describe some earlier integration experience in [48], and we present here extended work with other Cloud platforms such as Eucalyptus and Amazon EC2.

Please cite this article in press as: Y. Zhao, et al., Enabling scalable scientific workflow management in the Cloud, Future Generation Computer Systems (2014), http://dx.doi.org/10.1016/j.future.2014.10.023

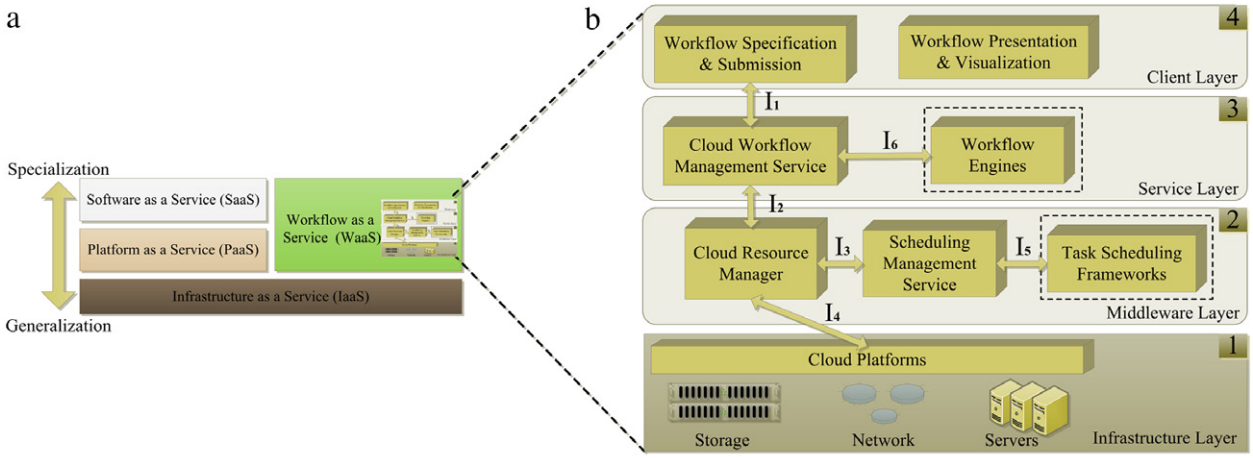


Fig. 4. The service framework.

4.1. 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" resource provider to an SWFMS, then tasks in a workflow can be dispatched 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 reference service framework that addresses the above mentioned challenges 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. 4(b), the service framework includes 4 layers, 8 components and 6 interfaces. Fig. 4(a) 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 layers, because our proposed service framework can also be applied to provide workflow platform as a service for related scientists.

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 and Submission and the Workflow Presentation and Visualization subsystem. The service architecture 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.

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.

4.2. The integration framework

Based on the service framework, we propose an end-to-end integration framework that covers all the major aspects involved in the integration, including a client side workflow submission tool, a Cloud workflow management service that accepts the submissions, a Cloud Resource Manager (CRM), which is a central component of the integration that accepts resource requests from the workflow service and dynamically instantiates a virtual cluster, and a cluster monitoring service that monitors the health of the acquired Cloud resources. The architecture of the integration is shown in Fig. 5. The Cloud workflow service accepts workflow submissions from the client tool, and makes resource requests to the Cloud resource manager, which in turn provisions a virtual cluster on-demand and also deploys the Falkon [20,49] execution service into the cluster. Falkon is a light-weight task execution service for optimized task throughput and resource efficiency delivered by a streamlined dispatcher, a dynamic resource provisioner. Individual jobs from the workflow service are then passed onto the Falkon service for parallel execution within the virtual cluster, and results delivered back to the workflow service. A common API is defined in between the CRM and the underlying Cloud platform, so that various Iaas Cloud platforms can be plugged in. We have firstly chosen to implement the interface with the OpenNebula Cloud platform to manage Cloud datacenter resources such as servers, network and storage. In addition, we firstly choose to integrate the Swift scientific workflow system with IaaS platforms. Swift is a parallel programming system for rapid and reliable specification, execution, and management of large-scale science and engineering workflows. Moreover, The Swift workflow system has flexible interfaces for developers to customize and can be easily mapped into our architecture.

4.3. The OpenNebula Cloud platform

OpenNebula is a full set of open-source toolkit to build IaaS private, public and hybrid Clouds, and a modular system that can implement a variety of Cloud architectures. OpenNebula orchestrates storage, network, virtualization, monitoring, and security

Please cite this article in press as: Y. Zhao, et al., Enabling scalable scientific workflow management in the Cloud, Future Generation Computer Systems (2014), http://dx.doi.org/10.1016/j.future.2014.10.023

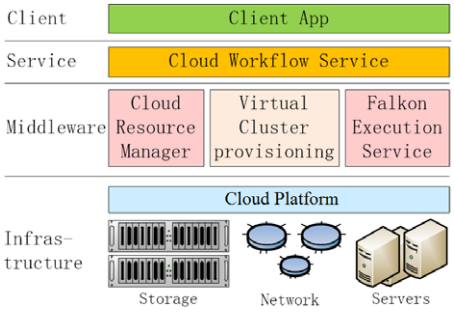


Fig. 5. The integration architecture.

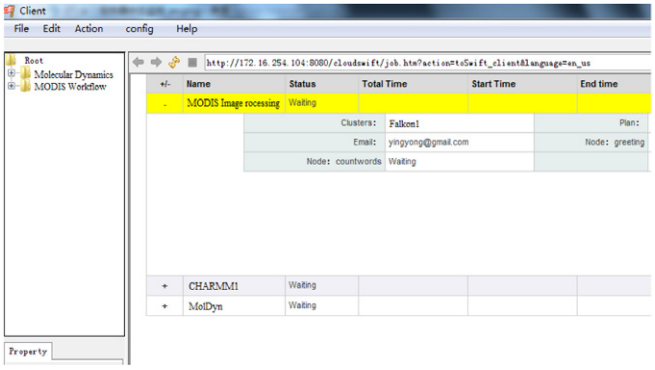


Fig. 6. The Client Submission Tool.

technologies to deploy multi-tier services [50,51] as virtual machines on distributed infrastructures, combining both datacenter resources and remote Cloud resources, according to allocation policies.

The OpenNebula internal architecture can be divided into three layers: *Tools*, *Core*, and *Drivers*.

Tools: This layer contains tools distributed with OpenNebula, such as the CLI, the scheduler, the libvirt API implementation or the Cloud RESTful interfaces, and also third party tools that can be easily created using the XML-RPC interface or the OpenNebula client API.

Core: The core consists of a set of components to control and monitor virtual machines, virtual networks, storage and hosts. The management of VMs, storage devices and virtual network is implemented in this layer by invoking a suitable driver.

Drivers: This layer is responsible for directly interacting with specific middleware (e.g. virtualization hypervisor, file transfer mechanisms or information services). It is designed to plug-in different virtualization, storage and monitoring technologies and Cloud services into the core.

We choose OpenNebula for our implementation because it has a flexible architecture and is easy to customize, and also because it provides a set of tools and service interfaces that are handy for integration. Other Cloud platforms can be integrated in similar means. In the following, we describe the core components in the integration and the interactions among them.

4.4. Key components

4.4.1. The Client Submission Tool

The Client Submission Tool (illustrated in Fig. 6) is a standalone Java application that provides an IDE for workflow development, and allows users to edit, compile, run and submit SwiftScripts. Scientists and application developers can write their scripts in this environment and also test run their workflows on local host, before they make final submissions to the Swift Cloud service to run in the Cloud. For submission, it provides multiple submission options: execute immediately, execute at a fixed time point, or execute recurrently (per day, per week etc.).

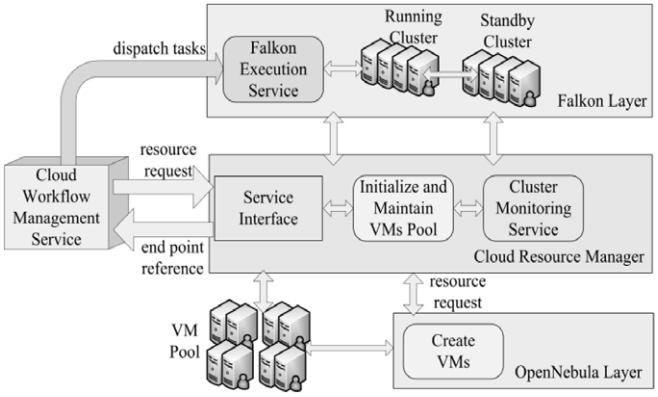


Fig. 7. System control diagram.

4.4.2. The Cloud workflow management service

As shown in Fig. 7, one of the key components of the system is the Cloud workflow management service that acts as an intermediary between the workflow client and the backend Cloud Resource Manager. The service has a Web interface for configuration of the service, the resource manager and application environments. It supports the following functionalities: SwiftScript programming, SwiftScript compilation, workflow scheduling, resource acquisition, and status monitoring. In addition, the service also implements fault-tolerance mechanism.

4.4.3. The Cloud Resource Manager

The Cloud Resource Manager accepts resource requests from the Cloud workflow management service, and is in charge of interfacing with the underlying Cloud platform (e.g. OpenNebula) and provisioning Falkon virtual clusters dynamically to the workflow service. In addition, it also monitors the virtual clusters. The process to start a Falkon virtual cluster is as follows:

(1) CRM provides a service interface to the workflow service, and the latter makes a resource request to CRM.
(2) CRM initializes and maintains a pool of virtual machines, the number of virtual machines in the pool can be set via a config file, Ganglia is started on each virtual machine to monitor CPU, memory and I/O.
(3) Upon a resource request from the workflow service, the CRM fetches a VM from the VM pool and starts the Falkon service in that VM. The CRM fetches another VM and starts the Falkon worker in that VM, which is registered to the Falkon service. CRM repeats fetching additional VMs and starting them until all the Falkon workers are started and registered. If the VMs in the pool are not enough, then CRM will make resource request to the underlying OpenNebula platform to create more VM instances.
(4) CRM returns the end point reference of the Falkon server to the workflow service, and the workflow service can now dispatch tasks to the Falkon execution service.
(5) CRM starts the Cluster Monitoring Service to monitor the health of the Falkon virtual cluster. The monitoring service checks heartbeat from all the VMs in the virtual cluster, and will restart a VM if it becomes irresponsive. Upon unrecoverable failure, or if the Falkon service fails, a new VM will be allocated and the Falkon service started, and all the workers would re-register to the new service. VMs running Falkon workers need only reconfigure their workers, such as shutting them down and restarting with the new service end-points.

Please cite this article in press as: Y. Zhao, et al., Enabling scalable scientific workflow management in the Cloud, Future Generation Computer Systems (2014), http://dx.doi.org/10.1016/j.future.2014.10.023

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

(1) If the number is more than requested, then the surplus workers are de-registered and put into the VM pool.
(2) If the number is less than required, then additional VMs will be pulled from the VM pool to create more workers.

As for the management of VM images, VM instances, and VM network, CRM interacts with and relies on the underlying OpenNebula Cloud platform. Our resource provisioning approach takes into consideration not only the dynamic creation and deployment of a virtual cluster with a ready-to-use execution service, but also efficient instantiation and re-use of the virtual cluster, as well as the monitoring and recovery of the virtual cluster. Next, we demonstrate the capability and efficiency of our integration using a small scale experiment setup.

4.5. Discussion

Through the introduction of the reference service framework and the implementation of different modules that can be mapped into the proposed framework, there are three advantages: (1) proposing a framework to bridge various SWFMSs with multiple heterogeneous Cloud environments; (2) breaking the limitations that a specific SWFMS is bound to a particular Cloud environment; (3) providing both practical and reference values to researchers who are devoted to the study of running scientific workflows in Clouds, so that they can contribute and share components designed and implemented by the guidance of the service framework, which is beneficial to both the workflow and the Cloud computing communities.

The separation between different layers can help to isolate the unnecessary influence between each layer, enhance the reusability of each module. Within the integration framework, we define a series of interfaces to standardize the integration between different SWFMSs and Cloud platforms. Based on the framework, every module can be replaced by its counterpart, as long as it implements the interfaces defined to interact with the other modules.

As we have discussed in the Challenges section, there are many major obstacles to the adaptation and running of scientific workflows on the Cloud. To address the architectural challenges, we take the "Workflow-Management-Layer-in-the-Cloud" approach for integration, which requires minimal configuration at the client side and supports easy deployment with virtualization techniques. We introduce the Cloud Resource Manager component to provide resource provision as a service for SWFMSs and deal with the compatibility and impedance problems that are introduced by different Cloud providers and heterogeneous implementations. As for dynamic scale-out and scale-in, we have discussed the resource provisioning, the various allocation and de-allocation policies, and how dynamic and adaptive provisioning can be adopted in light of varying workloads in our previous papers [52,53]. We can couple mechanisms similar to the dynamic resource provisioning (DRP) mechanism built in Falkon to enable scientific workflows to dynamically acquire/dismiss computing resources based on execution time and queue situation.

In addition, we utilize the access control mechanism to ensure security for Cloud-based SWFMSs, and we address data management challenges both inside and outside the SWFMSs, involving the Infrastructure layer, Middleware layer and Service layer. To address the language challenges, we adopt SwiftScript to serve as a general purpose coordination language, where existing applications can be invoked without modification. The service management challenges such as service discovery, large input and output handling, data services, etc. can be addressed inside the SWFMSs.

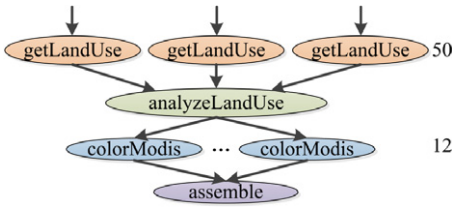


Fig. 8. MODIS image processing workflow.

5. Performance evaluation

In this section, we first demonstrate and analyze our integration with OpenNebula using a NASA MODIS image processing workflow. We then present the cluster provisioning results within the Eucalyptus Cloud platform and analyze the Montage Image Mosaic Workflow processing results in Amazon EC2. We also identify that distributed storage in Cloud platforms may be one of the key areas of improvement towards the better support of scientific workflows.

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 baseline 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 the following tables, we can clearly see that the recycling mechanism is efficient when initializing a cluster based on the "standby" Falkon cluster.

$$P_{TS} = [(T_B - T_C)/T_B] * 100\%. \quad (1)$$

5.1. The MODIS image processing workflow

The NASA MODIS dataset [54] we use is a set of satellite aerial data blocks, each block is of size around 5.5 MB, with digits indicating the geological feature of each point in that block, such as water, sand, green land, urban area, etc.

The workflow (illustrated in Fig. 8) takes a set of data 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.

Each machine in the experiment is configured with Intel Core i5 760 with 4 cores at 2.8 GHz, 4 GB memory, 500 GB HDD, and connected with Gigabit Ethernet LAN. The operating system is Ubuntu 10.04.1, installed with OpenNebula 2.2. The configuration for each VM is 1 core, 1.5 GB memory, 20 GB 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, we use the small set of resources in order to reach resource limit easily, nevertheless, they are enough for other performance tests.

In our experiment, we control the workload by changing the number of input data blocks and the resource required. We run two types of experiments:

(1) *The serial submission experiment.*
(2) *The different number of data blocks experiment.*

Please cite this article in press as: Y. Zhao, et al., Enabling scalable scientific workflow management in the Cloud, Future Generation Computer Systems (2014), http://dx.doi.org/10.1016/j.future.2014.10.023

Table 2
The baseline for cluster creation (OpenNebula).

Cluster size	Server initialization (s)	Worker registration (s)	Total time (s)
1	4.674	10.603	15.277
3	4.626	12.124	16.75
5	4.716	14.12	18.836
7	4.619	16.346	20.965
9	4.66	18.592	23.252

Table 3
Serial submission, decreasing required resources.

Cluster size	Time (s)	Baseline (s)	Time saved
5	17.961	18.836	–
3	4.138	16.75	75%
1	3.683	15.277	76%

In all the experiments, VMs are pre-instantiated and put in the VM pool. The time to instantiate a VM is around 42 s and this does not change much for all the VMs created.

5.1.1. Serial submission

In the serial submission experiment, we first measure the baseline for server initialization time and worker registration time. We create a Falkon virtual cluster with 1 server, and varying number of workers, and we do not reuse the virtual cluster.

In Table 2, we can observe that the server initialization time is quite stable, around 4.7 s every time, and for worker parallel registration, the time increases slightly with the worker number.

Then, we submit a workflow after the previous one has finished to test virtual cluster recycling. In Table 3, the resources required are one Falkon server with 5 workers, one server with 3 workers and one server with 1 worker. As the "standby" Falkon cluster can be reused, for the second and third submissions, the server initialization and worker registration time is zero, and only the surplus workers need to de-register themselves. We can clearly see that the recycling mechanism is efficient.

5.1.2. Varying the data blocks

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 Table 2). 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.

5.2. Eucalyptus experiments

In this section, we show the integration results of using Eucalyptus instead of OpenNebula for resource provisioning and as the underlying Cloud platform. Considering the efficient and convenient services provided by FutureGrid[1] with Eucalyptus

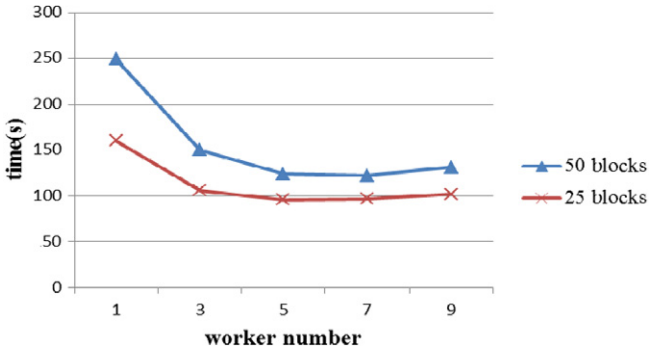


Fig. 9. Different input sizes.

Table 4
The baseline for virtual cluster creation (Eucalyptus).

Cluster size	Server initialization (s)	Worker registration (s)	Total time (s)
1	10.764	10.739	21.503
2	11.66	12.231	23.891
4	10.994	14.059	25.053
8	10.766	17.565	28.331
16	11.508	22.665	34.173
32	11.481	28.895	40.376

deployment, we choose FutureGrid as the experiment environment. 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 FutureGrid Project makes it possible for researchers to tackle complex research challenges in computer science related to the use and security of grids and clouds. These include topics ranging from authentication, authorization, scheduling, virtualization, middleware design, interface design, and cyber security, to the optimization of grid-enabled and cloud-enabled computational schemes for researchers in astronomy, chemistry, biology, engineering, atmospheric science and epidemiology.

In addition, the Eucalyptus API is compatible with Amazon EC2 so the implementation can easily support Amazon EC2 Cloud. We measure the performance to establish a baseline for resource provisioning and Cloud resource management overhead in the science Cloud environment.

5.2.1. Experiment configuration

The instance type used in our experiment is m1.small: 1 CPU Unit, 1 CPU Core and 500 MB Memory. All the instances use Ubuntu Server 12.04 as the operating system. In all the experiments, VM instances are pre-launched using created images and we initialize a resource pool with 33 instances.

5.2.2. Framework overhead evaluation

In the overhead evaluation experiment, we measure the server initialization time and worker registration time to compare with those in the OpenNebula setting.

In Table 4, we observe the time to create a Falkon server and start the service is around 11 s, much longer than that in Table 2. We attribute this to the m1.small configuration. Worker registration takes more time than in Table 2 due to slower network connection. 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 Table 2.

Then we measure the server initialization and worker registration time of one Falkon cluster consisting of one server and 32 workers. We submit requests with exponentially decreasing

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

Table 5
Serial submission, decreasing resource required.

Cluster size	Time (s)	Baseline (s)	Time saved
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%

Table 6
Serial submission, increasing resource required.

Cluster size	Time (s)	Baseline (s)	Time saved
1	21.98	21.503	–
2	11.16	23.891	53%
4	12.776	25.053	49%
8	13.81	28.331	51%
16	17.322	34.173	49%
32	22.15	40.376	45%

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. The results are shown in Table 5.

In Table 6, we measure the server initialization and worker registration time of a Falkon cluster starting from one server and one worker. Then we expand the cluster size exponentially by adding 1 worker → 2 workers → 4 workers → 8 workers → 16 workers into the cluster.

As shown in Table 6, we can see that although the worker number increases exponentially, the time rises almost linearly. The reason is that the workers can simultaneously register to the already existing server. The time cost to register each individual worker is similar to that in Table 4.

In Table 7, we first request a virtual cluster with 1 server and 32 workers, we then make 5 parallel requests for virtual clusters with 1 server and 5 workers. According to the cluster reuse mechanism, one of the clusters can be created based on available cluster, while the other 4 are created on-demand. In this case, it is much faster to de-register the surplus workers than to create the server from scratch.

From the experiment results in FutureGrid, we can observe that, compared with the local cluster based OpenNebula Cloud environment, within the production scientific Grid/Cloud platform, with a relatively larger resource provisioning setting, although the provisioning overhead is larger than in the local setting, the virtual cluster recycling mechanism works in similar pattern and efficiency.

5.3. Amazon EC2 experiments

In this section, we show the results of using Amazon EC2 as the resource provisioner, and use a Montage image mosaic workflow [28] for illustration. The workflow processes 2MASS nebula graph. The region size of the graph is 0.5, and the image data are divided into 18 FITS images with size of 2 MB in each survey band (H, J and Ks), the number of input image files to the workflow can vary from hundreds to tens of thousands. With these results, we demonstrate that our integration framework supports not only both research Clouds and commercial Clouds, but also a variety of scientific workflows.

5.3.1. The 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. 10: the left side shows the workflow stages

Table 7
Serial submission, mixed resource required.

Cluster size	Time (s)	Baseline (s)	Time saved
32	39.98	21.503	–
5	12.866	24.891	48%
5	25.514	24.891	–
5	24.302	24.891	–
5	24.987	24.891	–
5	25.62	24.891	–

for generating the mosaic of 3 images, and the right side shows the multiple stages (in ellipse) and the corresponding input and output files (in parallelogram) and their dependencies. 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.

There are two Amazon instance types used in our experiment. Falkon server and Swift server use the same configuration: instance type is c1.medium with 5 CPU Units, 2 CPU Cores and 1.7 GB memory. Falkon worker is configured with instance type m1.small: 1 CPU Unit, 1 CPU Core and 1.7 GB memory. All the instances use Ubuntu Server 11.10 as the operating system and are in the same Security Group.

In the experiment in Amazon EC2, we measure the time to initialize the Falkon cluster and to process 2MASS nebula graph, we perform the measurement 10 times and take the average. As the overall test environment is in a commercially operated Cloud, there may appear a few unreasonable data which have been deemed outliers and excluded. In all the experiments, VM instances are pre-launched using AMIs.

5.3.2. Application evaluation

In this experiment, we submit a 2MASS nebula graph processing workflow to the swift service, which schedules and dispatches tasks to a number of Falkon workers through the Falkon server. We change the number of workers in the Falkon cluster and measure the time of the whole procedure except the cluster initialization time.

In Fig. 11, we can observe that with the increase of the number of workers, the montage workflow processing time decreases accordingly. After the number of workers has reached 8, the time cost decreases slowly. If we take into consideration both the montage processing time and the cluster creation time, the total time becomes longer when the number of workers reaches 20, then we will arrive at the same conclusion as summarized from Fig. 9. We should choose an appropriate cluster size to achieve resource utilization efficiency and performance/cost balance. The saturation at 8-node scales is due to the data-intensive nature of the application, and the inability for the storage system to keep up with the demand. We believe that distributed storage in Cloud platforms is perhaps one of the key areas of improvement towards the better support of scientific workflows.

Please cite this article in press as: Y. Zhao, et al., Enabling scalable scientific workflow management in the Cloud, Future Generation Computer Systems (2014), http://dx.doi.org/10.1016/j.future.2014.10.023

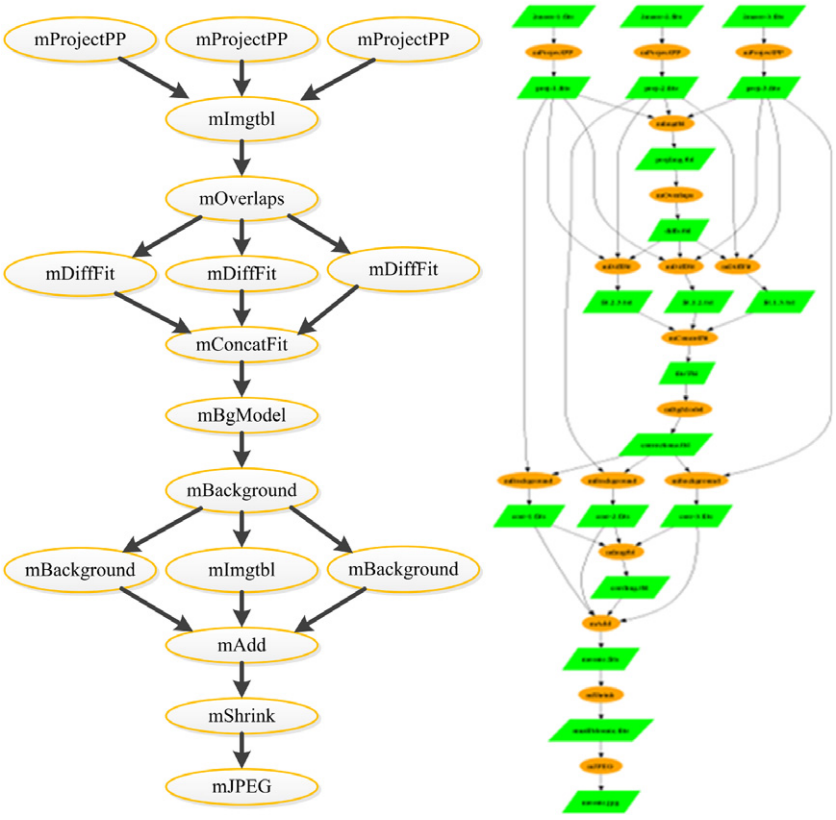


Fig. 10. The montage workflow.

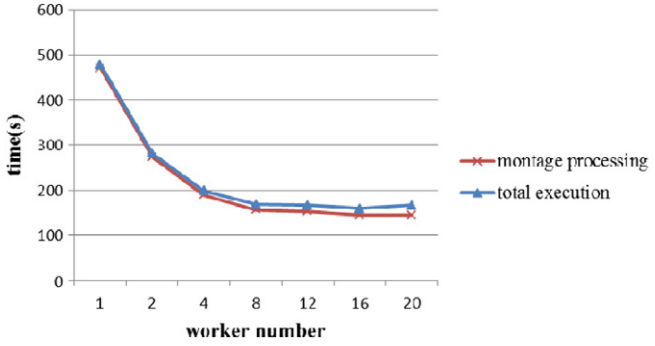


Fig. 11. Montage processing.

6. On-going work and research directions

The proposed framework is open, flexible and expandable. The separation between different layers makes it easy to introduce new and mature techniques and components into our framework to improve the performance and functionality. In this section, we would like to introduce our on-going work and present some research directions.

OpenStack integration: Currently, we are cooperating with the Joint Lab for Medical Physics of Sichuan Provincial People's Hospital, using our proposed framework to process massive medical images. We use Openstack as the underlying Cloud platform, which is the most popular open-source Cloud computing platform at present. The platform consists of a series of interrelated projects that control pools of processing, storage, and networking resources throughout a datacenter, which can be managed or provisioned through a Web-based dashboard, command-line tools, or a RESTful API. The OpenStack APIs are compatible with Amazon EC2/S3 and thus client applications written for Amazon Web Services can be used with OpenStack with minimal porting effort. We have defined clear interfaces in the framework, so that the Openstack Cloud platform can be easily mapped into the framework, as long as we implement the defined interfaces to communicate with the other components in the framework.

Auto-scaling: To minimize cost and meet application deadlines in Cloud workflows, we are investigating the auto-scaling mechanism, which can adjust the number of workers automatically according to workflow workload. As we have discussed above, we may use the interfaces defined in the task scheduling framework to monitor the task queue during the execution, and dynamically acquire/dismiss computing resources based on execution time and queue situation.

Load balancing: We are also integrating work stealing strategy in the task scheduling framework that can improve workflow task scheduling efficiency and load balance. We have improved three dynamic load balancing algorithms, including addition strategy, multiplication strategy, and dichotomy strategy [55]. Based on the above work-stealing strategies, we are investigating the prefetching mechanism to improve resource utilization and task scheduling throughput.

Dynamic network provisioning: We have seen recent advances in enabling on-demand network circuits in the national and international backbones coupled with Software Defined Networking (SDN) advances like OpenFlow and programmable edge technologies like OpenStack [56]. Integrating SDN techniques into our framework enables network control to become directly programmable and the bandwidth-provisioned high-speed circuits to be allocated dynamically, which can increase the ability of Cloud workflows to access and stage large datasets from remote data repositories or to move computation to remote sites and access data stored locally.

Cloud orchestration: With the proposed framework, we can provide scientific workflow platform as a service to researchers

Please cite this article in press as: Y. Zhao, et al., Enabling scalable scientific workflow management in the Cloud, Future Generation Computer Systems (2014), http://dx.doi.org/10.1016/j.future.2014.10.023

and workflow developers. However, the deployment of such a platform still requires a lot of efforts and specific techniques. We can apply Cloud orchestration to realize the automatic deployment and management of cloud workflow platform. To standardize the processes and operations, we are looking at the TOSCA (the Topology and Orchestration Specification for Cloud Applications) standard to model the components, relationships between components, and components management.

7. Conclusions

As more and more scientific communities and applications are adopting Cloud platforms, it is important to migrate SWFMSs into Clouds to take advantage of both convenient and powerful workflow management and Cloud scalability, and to handle the ever increasing demand of big data applications. Cloud offers unprecedented scalability to workflow systems, and will 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 nature of resource allocation on the Cloud will also help improve resource utilization and user experience. We coin the term "Cloud Workflow" and analyze the major obstacles to the adaptation and running of scientific workflows on the Cloud. Based on the service framework, we present our early effort in designing an integration framework for scientific workflow management in the Cloud, of which a Cloud workflow management service, a Cloud resource manager, and a cluster monitoring service are core components. We also conduct a set of experiments in OpenNebula as well as in Eucalyptus and Amazon EC2 to measure resource provisioning and management efficiency and to find the cost/performance balance.

Acknowledgments

This work was supported by National Natural Science Foundation of China No. 61272528, No. 61034005, and the Central University Fund (ID-ZYGX2013J073).

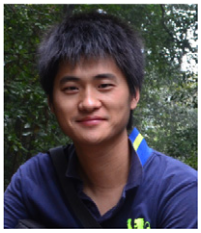
References

[1] I. Foster, Y. Zhao, I. Raicu, S. Lu, Cloud computing and grid computing 360-degree compared, in: IEEE Grid Computing Environments, GCE08, 2008, Co-located with IEEE/ACM Supercomputing 2008, Austin, TX, pp. 1–10.
[2] Hadoop, 2014 [Online]. Available: http://hadoop.apache.org/.
[3] OpenNebula, 2014 [Online]. Available: http://www.OpenNebula.org.
[4] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, D. Zagorodnov, The eucalyptus open-source cloud-computing system, in: 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID'09, 2009, pp. 124–131.
[5] K. Keahey, T. Freeman, Contextualization: providing one-click virtual clusters, in: eScience 2008, Indianapolis, IN, 2008, pp. 301–308.
[6] Openstack, 2014 [Online]. Available: http://www.openstack.org.
[7] G. Bell, T. Hey, A. Szalay, Beyond the data deluge, Science 323 (5919) (2009) 1297–1298.
[8] Yong Zhao, Youfu Li, Shiyong Lu, Ioan Raicu, Cui Lin, Devising a cloud scientific workflow platform for big data, in: IEEE International Symposium on Scientific Workflows and Big Data Science, SWF, 2014.
[9] Y. Zhao, X. Fei, I. Raicu, S. Lu, Opportunities and challenges in running scientific workflows on the cloud, in: IEEE International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC, 2011, pp. 455–462.
[10] C. Lin, S. Lu, X. Fei, A. Chebotko, D. Pai, Z. Lai, F. Fotouhi, J. Hua, A reference architecture for scientific workflow management systems and the VIEW SOA solution, IEEE Trans. Serv. Comput. (TSC) 2 (1) (2009) 79–92.
[11] Yong Zhao, Shiyong Lu, A logic programming approach to scientific workflow provenance querying, in: Proc. of the 2008 International Provenance and Annotation Workshop (IPAW 2008), Salt Lake City, Utah, in: Lecture Notes in Computer Science, vol. 5272, 2008, pp. 31–44.
[12] Rob Pike, Sean Dorward, Robert Griesemer, Sean Quinlan, Interpreting the data: parallel analysis with Sawzall, Sci. Program. 13 (4) (2005) 277–298.
[13] C. Olston, B. Reed, U. Srivastava, R. Kumar, A. Tomkins, Pig Latin: a not-so-foreign language for data processing, in: SIGMOD 2008.
[14] Y. Yu, M. Isard, D. Fetterly, M. Budiu, U. Erlingsson, P.K. Gunda, J. Currey, DryadLINQ: a system for general-purpose distributed data-parallel computing using a high-level language, in: Symposium on Operating System Design and Implementation, OSDI, San Diego, CA, December 8–10, 2008.
[15] 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, Honolulu, Hawaii, USA, July 2008, pp. 335–342.
[16] Andrey Kashlev, Shiyong Lu, Artem Chebotko, Coercion approach to the shimming problem in scientific workflows, in: Proc. of the IEEE International Conference on Services Computing, SCC, Santa Clara, CA, USA, 2013.
[17] Cui Lin, Shiyong Lu, Xubo Fei, Darshan Pai, Jing Hua, A task abstraction and mapping approach to the shimming problem in scientific workflows, in: IEEE International Conference on Services Computing, SCC, Bangalore, India, 2009, pp. 284–291.
[18] W. Tan, K. Chard, D. Sulakhe, R. Madduri, I. Foster, S.S. Reyes, C. Goble, Scientific workflows as services in caGrid: a Taverna and gRAVI approach, in: ICWS 2009, pp. 413–420.
[19] C. Vecchiola, S. Pandey, R. Buyya, High-performance cloud computing: a view of scientific applications, in: International Symposium on Parallel Architectures, Algorithms, and Networks, 2009, pp. 4–16.
[20] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, M. Wilde, Falkon: a fast and light-weight tasK executiON framework, in: IEEE/ACM SuperComputing 2007, pp. 1–12.
[21] A. Chebotko, S. Lu, S. Chang, et al., Secure abstraction views for scientific workflow provenance querying, IEEE Trans. Serv. Comput. 3 (4) (2010) 322–337.
[22] Z. Yang, S. Lu, P. Yang, Itinerary-based access control for mobile tasks in scientific workflows, in: Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on IEEE, Vol. 2, 2007, pp. 506–511.
[23] A. Chebotko, C. Lin, X. Fei, et al., VIEW: a VIsual sciEntificWorkflow management system, in: Services 2007 IEEE Congress on, IEEE, 2007, pp. 207–208.
[24] 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 Rec. 34 (3) (2005) 37–43.
[25] M. Palankar, A. Iamnitchi, M. Ripeanu, S. Garfinkel, Amazon S3 for science grids: a viable solution?, in: Proceedings of the 2008 International Workshop on Data-aware Distributed Computing, DADC'08, 2008, pp. 55–64.
[26] D. Oliveira, K. Ocaña, E. Ogasawara, J. Dias, F. Baião, M. Mattoso, A performance evaluation of X-ray crystallography scientific workflow using SciCumulus, in: IEEE CLOUD 2011, pp. 708–715.
[27] L. Wang, J. Tao, M. Kunze, A.C. Castellanos, D. Kramer, W. Karl, Scientific cloud computing: early definition and experience, in: 10th IEEE International Conference on High Performance Computing and Communications, HPCC'08, 2008, pp. 825–830.
[28] E. Deelman, G. Singh, M. Livny, B. Berriman, J. Good, The cost of doing science on the cloud: the Montage example, in: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, SC'08, Piscataway, NJ, USA, 2008, pp. 50:1–50:12.
[29] S. Srirama, O. Batrashev, P. Jakovits, E. Vainikko, Scalability of parallel scientific applications on the cloud, Sci. Program. J. 19 (2–3) (2011) (Special Issue on Science-Driven Cloud Computing).
[30] M. Parashar, H. Kim, Autonomic management of applications workflows on hybrid computing infrastructure, Sci. Program. J. 19 (2–3) (2011) (Special Issue on Science-Driven Cloud Computing).
[31] C. Hoffa, G. Mehta, T. Freeman, et al., On the use of cloud computing for scientific workflows, in: eScience, 2008. eScience'08. IEEE Fourth International Conference on, IEEE, 2008, pp. 640–645.
[32] G. Juve, E. Deelman, K. Vahi, et al., Scientific workflow applications on Amazon EC2, in: E-Science Workshops, 2009 5th IEEE International Conference on, IEEE, 2009, pp. 59–66.
[33] G. Juve, E. Deelman, K. Vahi, et al., Data sharing options for scientific workflows on Amazon EC2, in: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE Computer Society, 2010, pp. 1–9.
[34] G. Juve, M. Rynge, E. Deelman, et al., Comparing FutureGrid, Amazon EC2, and open science grid for scientific workflows, Comput. Sci. Eng. 15 (4) (2013) 20–29.
[35] L. Ramakrishnan, C. Koelbel, Y.-S. Kee, R. Wolski, D. Nurmi, D. Gannon, G. Obertelli, A. YarKhan, A. Mandal, T.M. Huang, K. Thyagaraja, D. Zagorodnov, VGrADS: enabling e-science workflows on grids and clouds with fault tolerance, in: Proc. Conf. High Performance Computing Networking, Storage and Analysis, SC'09, No. 47, 2009.
[36] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. Pocock, P. Li, T. Oinn, Taverna: a tool for building and running workflows of services, Nucleic Acids Res. 34 (Web Server Issue) (2006) 729–732.
[37] 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, Concurr. Comput.: Pract. Exper. 18 (10) (2006) 1039–1065 (Special Issue: Workflow in Grid Systems).
[38] J. Freire, C.T. Silva, S.P. Callahan, E. Santos, C.E. Scheidegger, H.T. Vo, Managing rapidly-evolving scientific workflows, in: Provenance and Annotation of Data, in: Lecture Notes in Computer Science, vol. 4145, 2006, pp. 10–18. http://dx.doi.org/10.1007/11890850_2.
[39] E. Deelman, et al., Pegasus: a framework for mapping complex scientific workflows onto distributed systems, Sci. Program. 13 (3) (2005) 219–237.

Please cite this article in press as: Y. Zhao, et al., Enabling scalable scientific workflow management in the Cloud, Future Generation Computer Systems (2014), http://dx.doi.org/10.1016/j.future.2014.10.023

[40] Y. Zhao, M. Hategan, B. Clifford, I. Foster, G.v. Laszewski, I. Raicu, T.S. -Praun, M. Wilde, Swift: fast, reliable, loosely coupled parallel computation, in: IEEE Workshop on Scientific Workflows 2007, pp. 199–206.
[41] M. Woitaszek, J. Dennis, T. Sines, Parallel high-resolution climate data analysis using swift, in: 4th Workshop on Many-Task Computing on Grids and Supercomputers, 2011.
[42] K. Damkliang, P. Tandayya, T. Phusantisampan, et al., Taverna workflow and supporting service for single nucleotide polymorphisms analysis, in: Information Management and Engineering, 2009. ICIME'09. International Conference on, IEEE, 2009, pp. 27–31.
[43] J. Zhang, P. Votava, T.J. Lee, et al., Bridging vistrails scientific workflow management system to high performance computing, in: Services (SERVICES), 203 IEEE Ninth World Congress on, IEEE, 2013, pp. 29–36.
[44] J. Zhang, Ontology-driven composition and validation of scientific grid workflows in Kepler: a case study of hyperspectral image processing, in: Grid and Cooperative Computing Workshops, 2006. GCCW'06. Fifth International Conference on, IEEE, 2006, pp. 282–289.
[45] M. Islam, A.K. Huang, M. Battisha, et al., Oozie: towards a scalable workflow management system for hadoop, in: Proceedings of the 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies, ACM, 2012, p. 4.
[46] M. Janetschek, S. Ostermann, R. Prodan, Bringing scientific workflows to Amazon SWF, in: Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference on, IEEE, 2013, pp. 389–396.
[47] 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 Comput. 42 (11) (2009) 50–60 (Special Issue on Extreme Scale Computing).
[48] Y. Zhao, Y. Zhang, W. Tian, R. Xue, C. Lin, Designing and deploying a scientific computing cloud platform, in: ACM/IEEE 13th International Conference on Grid Computing, 2012, pp. 104–113.
[49] I. Raicu, Y. Zhao, I. Foster, A. Szalay, Accelerating large-scale data exploration through data diffusion, in: International Workshop on Data-Aware Distributed Computing 2008, Co-locate with ACM/IEEE International Symposium High Performance Distributed Computing, HPDC, 2008, pp. 9–18.
[50] R. Moreno-Vozmediano, R.S. Montero, I.M. Llorente, Multi-cloud deployment of computing clusters for loosely-coupled MTC applications, IEEE Trans. Parallel Distrib. Syst. 22 (6) (2011) 924–930.
[51] R.S. Montero, R. Moreno-Vozmediano, I.M. Llorente, An elasticity model for high throughput computing clusters, J. Parallel Distrib. Comput. 71 (6) (2011) 750–757.
[52] I. Raicu, Y. Zhao, C. Dumitrescu, et al. Dynamic resource provisioning in grid environments, 2007.
[53] I. Raicu, I.T. Foster, Y. Zhao, et al., The quest for scalable support of data-intensive workloads in distributed systems, in: Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing, ACM, 2009, pp. 207–216.
[54] NASA MODIS dataset, 2014 [Online]. Available: http://modis.gsfc.nasa.gov/.
[55] Work-stealing strategies technical report, 2014 [Online]. Available: http://www.cloud-uestc.cn/projects/serviceframework/resource/documents/technical-report_work-stealing.pdf.
[56] P. Ruth, A. Mandal, Y. Xin, et al., Dynamic network provisioning for data intensive applications in the cloud, in: E-Science (e-Science), 2012 IEEE 8th International Conference on, IEEE, 2012, pp. 1–2.

Yong Zhao is a professor at the School of Computer Science and Engineering, University of Electronic Science and Technology of China. Before joining the university, he worked at Microsoft on Business Intelligence projects that leveraged Cloud storage and computing infrastructure. He obtained his Ph.D. in Computer Science from the University of Chicago under Dr. Ian Foster's supervision. He has published more than 30 papers in top journals and conferences, which are referenced more than 2000 times by other scholars and researchers. His research areas are in cloud computing, many-task computing and data intensive computing. He is a member of ACM, IEEE and CCF.



Youfu Li is a graduate student with the School of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interest is in Cloud computing, scientific workflows and real-time computing. He is a student member of the IEEE.

Ioan Raicu is an assistant professor in the Department of Computer Science at Illinois Institute of Technology, as well as a guest research faculty in the Maths and Computer Science Division at Argonne National Laboratory (ANL). He has received the prestigious NSF CAREER award (2011–2015) for his innovative work on distributed file systems for exascale computing. He obtained his Ph.D. in Computer Science from the University of Chicago under the guidance of Dr. Ian Foster in March 2009. He is particularly interested in resource management in large scale distributed systems with a focus on many-task computing, data intensive computing, cloud computing, grid computing, and many-core computing. He is a member of the IEEE and ACM.

Shiyong Lu is an associate professor in the Department of Computer Science at Wayne State University and the Director of the Big Data Research Laboratory. He received his Ph.D. in Computer Science from Stony Brook University in 2002. His research focuses on big data, scientific workflows, and data mining. He is an author of two books and over 100 papers. He is the founding chair of IEEE International Symposium of Scientific Workflows and Big Data Science since 2007. He is a Co-Editor-in-Chief of the International Journal of Cloud Computing and Services Science. He is a senior member of the IEEE.

Wenhong Tian is an associate professor at the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China. His research interest is in resource management and scheduling in Cloud datacenters. He is a member of the IEEE.

Heng Liu is a student with the School of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interest is in Cloud computing and Big Data.

Please cite this article in press as: Y. Zhao, et al., Enabling scalable scientific workflow management in the Cloud, Future Generation Computer Systems (2014), http://dx.doi.org/10.1016/j.future.2014.10.023