Migrating Scientific Workflow Management Systems from the Grid to the Cloud

Yong Zhao[1], Youfu Li[1], Ioan Raicu[2], Cui Lin[3], Wenhong Tian[1], Ruini Xue[1]

[1]School of Computer Science and Engineering, Univ. of Electronic Science and Technology of China, Chengdu, China

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

[3]Department of Computer Science, California State University, Fresno CA, USA

yongzh04@gmail.com, youfuli.fly@gmail.com, iraicu@cs.iit.edu, clin@csufresno.edu, tian_wenhong@uestc.edu.cn, xueruini@gmail.com

Abstract: Cloud computing is an emerging computing paradigm that can offer unprecedented scalability and resources on demand, and is gaining significant adoption in the science community. At the same time, scientific workflow management systems provide essential support and functionality to scientific computing, such as management of data and task dependencies, job scheduling and execution, provenance tracking, fault tolerance. Migrating scientific workflow management systems from traditional Grid computing environments into the Cloud would enable a much broader user base to conduct their scientific research with ever increasing data scale and analysis complexity. This paper presents our experience in integrating the Swift scientific workflow management system with the OpenNebula Cloud platform, which supports workflow specification and submission, on-demand virtual cluster provisioning, high-throughput task scheduling and execution, and efficient and scalable resource management in the Cloud. We set up a series of experiments to demonstrate the capability of our integration and use a MODIS image processing workflow as a showcase of the implementation.

Keywords: *Cloud workflow; Virtual Cluster Provisioning; Workflow-as-a-Service; Swift; OpenNebula*

1 Introduction

Scientific workflow management systems (SWFMS) have been proven essential to scientific computing as they provide functionalities such as workflow specification, process coordination, job scheduling and execution, provenance tracking [64], fault tolerance etc. SWFMS in fact represent a subset of Many-Task Computing (MTC) [61] workloads. MTC is reminiscent of High-Throughput Computing, but it differs in the emphasis of using many computing resources over short periods of time to accomplish many computational tasks (i.e. including both dependent and independent tasks), where the primary metrics are measured in seconds (e.g. FLOPS, tasks/s, MB/s I/O rates), as opposed to operations (e.g. jobs) per month. MTC denotes high-performance computations comprising multiple distinct activities, coupled via file system or memory-to-memory transfer operations. Tasks may be small or large, uniprocessor or multiprocessor, compute-intensive or data-intensive. The set of tasks may be static or dynamic, homogeneous or heterogeneous, loosely coupled or tightly coupled. The aggregate number of tasks, quantity of computing, and volumes of data may be extremely large [62]. MTC includes loosely coupled applications that are generally communication-intensive but not naturally expressed using standard message passing interface commonly found in HPC, drawing attention to the many computations that are heterogeneous but not "happily" parallel. [63] There are unprecedented challenges raised for traditional scientific workflows, as the data scale and computation complexity are growing exponentially. The ETL (Extraction-Transformation-Loading), storage, retrieval, analysis and application upon the huge amounts of data are beyond the capability of traditional data processing infrastructures. The community has coined this as Big Data, and it is often associated with the Cloud Computing paradigm.

As an emerging computing paradigm, Cloud computing [6] is gaining tremendous momentum 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 [33], OpenNebula [1], Eucalyptus [34], Nimbus [22], and OpenStack [2], 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 Internet. For instance, with a credit card, you can get access to Amazon EC2 virtual machines immediately; 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 Cloud takes care of scaling out and in, and load 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.

Theoretically, to address the big data problems in the above scientific computing areas, scientists and application developers may simply refactor all the existing workflow applications into the Cloud computing paradigm, which sounds straightforward but in reality is impractical. As traditional scientific workflow applications have been mature during many years' development and always involve complicated application logic and consist of massive computing processes such as organization, distribution, coordination and parallel processing. Transforming these scientific workflows will not only cost scientists and developers much time, but also require manual handling of all the integration details with various underlying Cloud platforms.

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

This paper extends earlier work [12] in which we identified various challenges associated with migrating and adapting an SWFMS in the Cloud. In this paper, *we present an end-to-end approach that addresses the integration of Swift, an SWFMS that has broad application in Grids and supercomputers, with the OpenNebula Cloud 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.

This paper's major contributions are:

1. *We analyze the challenges for traditional scientific workflows in the Grid environment, and proposed a structured approach to migrating a SWFMS into the Cloud.*
2. *We integrate Swift with OpenNebula, in order to coordinate and automate scientific analysis and discovery.*

3. *We propose a virtual cluster provisioning mechanism that could recycle Cloud virtual machine instances.*
4. *We present a use case as a showcase of the implementation.*

The rest of the paper is organized as follows: In the next section, we discuss the challenges of traditional scientific workflows and analyze the available solutions to the challenges. In the integration section, we introduce a service framework for the integration of SWFMSs and Cloud platforms and present our end-to-end integration of Swift and OpenNebula. In the performance evaluation section, we set up a series of experiments to analyze the integration and demonstrate the implementation using a NASA MODIS image processing workflow. In the related work section, we discuss related work in migrating scientific workflow management systems from the Grid to the Cloud. In the last section, we draw our conclusions and discuss future work.

2 Challenges and Available Solutions

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

2.1 Challenges for Traditional Scientific Workflows

Scientific workflow systems have been formerly applied over a number of execution environments such as workstations, clusters/Grids, and supercomputers. In contrast to Cloud environment, running workflows in these environments are facing a series of obstacles when dealing with big data problems [45], including data scale and computation complexity, resource provisioning, collaboration in heterogeneous environments, etc.

2.1.1 Data Scale and Computation Complexity

The execution of scientific workflows often consumes and produces huge amounts of distributed data objects. These data objects can be of primitive or complex types, files in different sizes and formats, database tables, or data objects in other forms. At present, the scientific community is facing a “data deluge” [7] coming from experiments, simulations, networks, sensors, and satellites, and the data that needs to be processed generally grows faster than computational resources and their speed. The data scale and managemnent in big data era are beyond the capability of traditional workflows as they depend on traditional infrastructure for resource provisioning, scheduling and computing. For example, in high energy physics, the Large Hadron Collider [4] at CERN can generate more than 100TB of collision data per second; In bioinformatics, GenBank[3], one of the largest DNA databases, already hosts over 120 billion bases, the European Molecular Biology and Bioinformatics Institute Laboratory (EMBL) hosts 14 PB of data, and the numbers are expected to double every 9-12 months.

In addition to data scale, science analysis and processing complexity is also growing exponentially. Scientists are now attempting calculations requiring orders of magnitude more computing and communication than was possible only a few years ago. For instance, in bioinformatics a protein simulation problem [29] involves running many instances of a structure prediction simulation, each with different random initial conditions and performs multiple rounds. Given a couple of proteins and parameter options, the simulation can easily scale up to 100,000 rounds. In cancer drug design, protein docking can involve millions of 3D structures and have a runtime up to tens of CPU years. To enable the storage and analysis of such large quantities of data and to achieve rapid turnaround, data and computation may need to be distributed over thousands or even tens of thousands of computation nodes.

2.1.2 Resource Provisioning

Resource provisioning represents the functionality and mechanism of allocating computing resource, storage space, network bandwidth, etc., to scientific workflows. As cluster/Grid environments are not adept at providing the workflows with smoothly dynamic resource allocation, the resource provisioned to a scientific workflow is fixed once the workflow has been deployed to execute, which may in return restrict the scale of science problems that can be handled by workflows. Moreover, the scale of resource is upbounded by the size of a dedicated resource pool with limited resource sharing extension in the form of virtual organizations. Meanwhile, the representation of resources in the context of scientific workflows is also bothering the scientists [46], as they must be able to recognize the supported types of resources and tools. For instance, the resource in Taverna is a web service which usually limits the use of many scientific resources that are not represented as web services.

To break through the limitations introduced by traditional resource provisioning strategy, some works have been focused on the approaches for automated provisioning, including the Context Broker [22] from the Nimbus project, which supported the concept of "one-click virtual cluster" that allowed clients to coordinate large virtual cluster launches in simple steps. The Wrangler system [23] was a similar implementation that allowed users to describe a desired virtual cluster in XML format, and send it to a web service, which managed the provisioning of virtual machines and the deployment of software and services. It was also capable of interfacing with many different Cloud resource providers.

2.1.3 Collaboration in Heterogeneous Environments

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

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

2.2 Moving Workflow Applications to the Cloud

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

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

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

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

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

4) Once we decide to migrate workflow applications to Cloud computing, we need to reconstruct the data being processed to be able to be stored in partitioned fashion, such as in GFS, or HDFS, so that the partitions can be operated in paral-

lel, which may introduce a tremendous amount of work to scientists and application developers.

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

2.3 Migrating Workflow Management into the Cloud

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

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

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

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

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

number of resources upfront. Cloud workflows can scale out and in dynamically, resulting in fast turn-around time for end users.

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

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

3 Integration of Swift and OpenNebula

In this section, we talk about our end-to-end approach in integrating Swift with the OpenNebula Cloud platform. Before we go into further details of the integration, we will first introduce the reference service framework that we propose to migrate scientific workflows to various Cloud platforms.

3.1 The Service Framework

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. 2, the service framework includes 4 layers, 7 components and 6 interfaces. Detailed description of the service framework is made public at our website[1].

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

We present a layered service framework for the implementation and application of integrating SWFMS into manifold Cloud platforms, which can also be ap-

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

plicable 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.

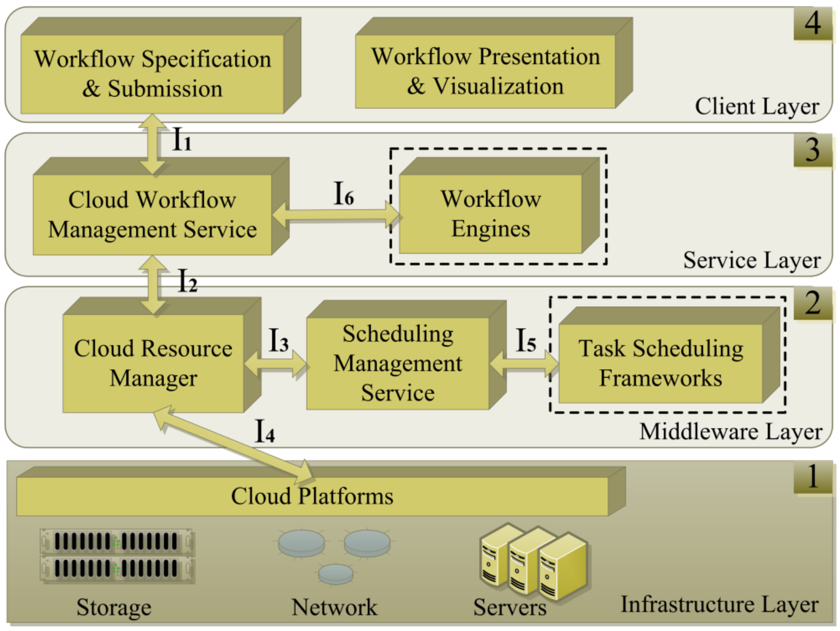

Workflow Specification & Submission
Workflow Presentation & Visualization
4
Client Layer
I1
3
Cloud Workflow Management Service
I6
Workflow Engines
Service Layer
I2
2
Cloud Resource Manager
I3
Scheduling Management Service
I5
Task Scheduling Frameworks
I4
Middleware Layer
1
Cloud Platforms
Storage
Network
Servers
Infrastructure Layer

Fig. 1. The Service Framework

3.2 Integration Architecture and Implementation

Based on the service framework, we devise an end-to-end integration approach that addresses the aforementioned challenges. We call it end-to-end because it 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) that accepts resource requests from the workflow service and dynamically instantiates a Falkon virtual cluster, and a cluster monitoring service that monitors the health of the acquired Cloud resources.

As illustrated in Fig. 3, the integration architecture consists of four layers. At the client layer, we provide a client-side development and submission tool for application specification and submission. At the service layer, a Cloud workflow service based on the Swift workflow management system [32] is presented as a gateway to the Cloud platform underneath. At the middleware layer, a few components are integrated seamlessly to bridge the gap between the service layer and the underlying infrastructure layer. The components include a Cloud resource manager, a virtual cluster provisioner, and a task execution service. 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 [27] execution service into the cluster. 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. At the infrastructure layer, we choose the OpenNebula Cloud platform to manage Cloud datacenter resources such as servers, network and storage.

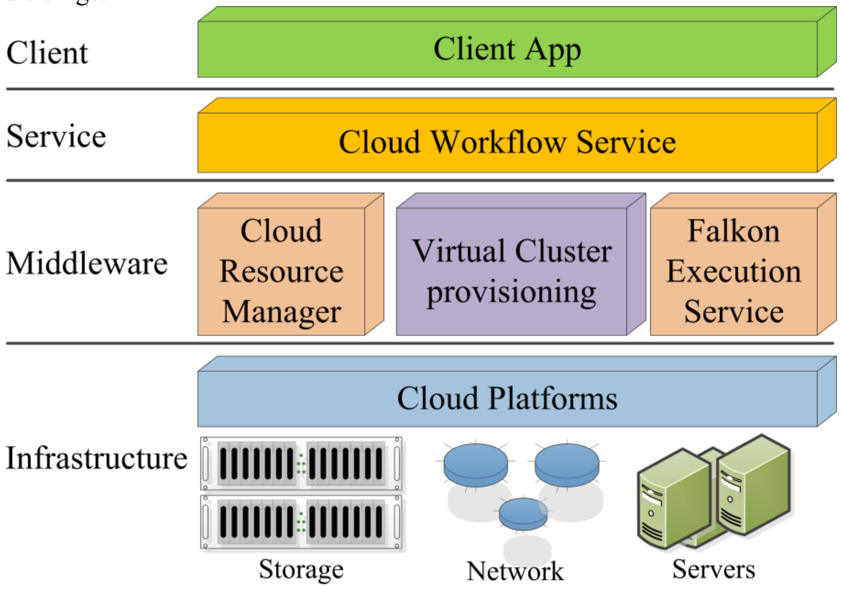


Fig. 2 The Integration Architecture

3.2.1 The Swift Workflow Management System

Swift is a system that bridges scientific workflows with parallel computing. It is a parallel programming tool 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 a simple scripting language called SwiftScript for concise specification of complex parallel computations based on dataset typing and iterations [31], 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. 4.

Computations are specified in SwiftScript, which has been shown to be simple yet powerful. SwiftScript programs are compiled into abstract computation plans, which are then scheduled for execution by the workflow engine onto provisioned resources. Resource provisioning in Swift is very flexible, tasks can be scheduled to execute on various resource providers, where the provider interface can be implemented as a local host, a cluster, a multi-site Grid, or the Amazon EC2 service.

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

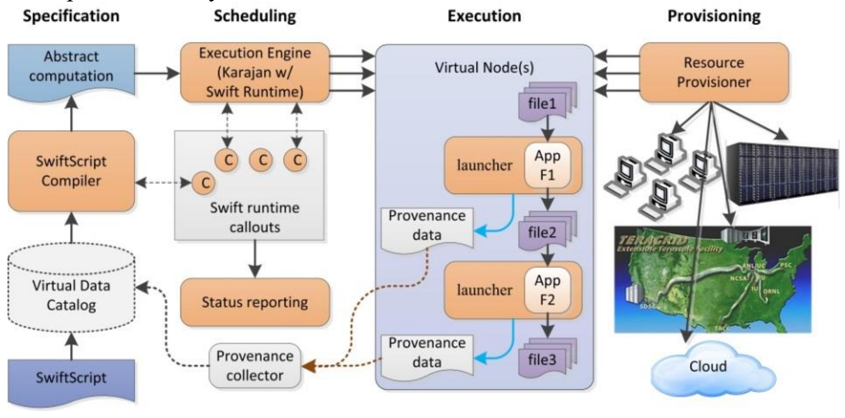


Fig. 3 Swift System Architecture

3.2.2 The OpenNebula Cloud Platform

We integrate Swift with the OpenNebula Cloud platform. 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 the integration. We have also integrated with other Cloud platforms such as Amazon EC2 and Eucalyptus in similar means.

OpenNebula is a fully open-source toolkit to build IaaS private, public and hybrid Clouds, and a modular system that can implement a variety of Cloud architectures and can interface with multiple datacenter services. OpenNebula orchestrates storage, network, virtualization, monitoring, and security technologies to deploy multi-tier services [38] [39] as virtual machines on distributed infrastructures, combining both datacenter resources and remote Cloud resources, according to allocation policies.

The OpenNebula internal architecture (as shown in Fig. 5) can be divided into three layers: *Drivers, Core and Tools* [40]:

1. *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.
2. *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.
3. *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.

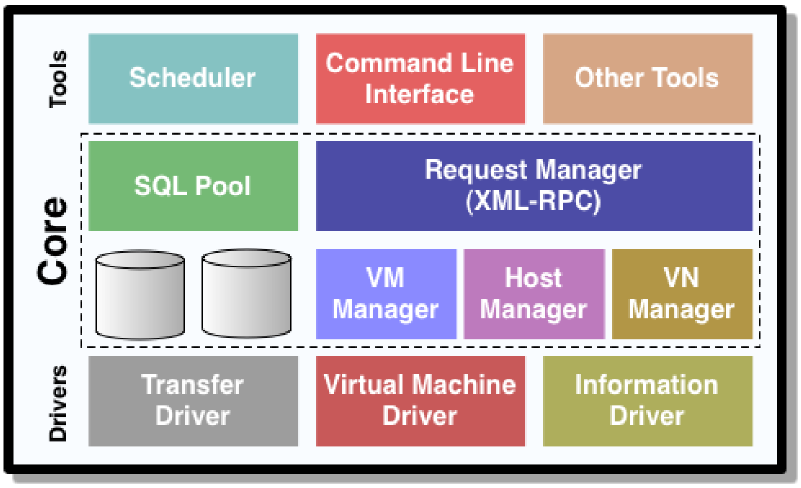
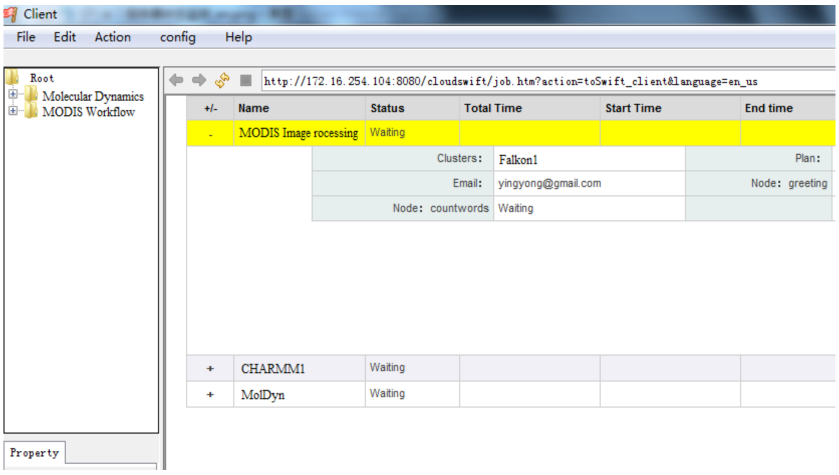


Fig. 4 The OpenNebula Architecture

3.2.3 Key Components

The client submission tool: The client submission tool 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.). We give a screenshot of the tool in Fig. 6, which shows the current status of workflows submitted to the Cloud service.

One of the key components of the system is the ***Swift 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. A screenshot of the service that visualizes workflow execution progress is shown in Fig. 7.



http://172.16.254.104:8080/cloudswift/job.htm?action=toSwift_client&language=en_us

+/-	Name	Status	Total Time	Start Time	End time
-	MODIS Image rocessing	Waiting			
+	CHARMM1	Waiting			
+	MolDyn	Waiting			

Clusters:	Falkon1	Plan:
Email:	yingyong@gmail.com	Node: greeting
Node: countwords	Waiting	

Fig. 5. The Client Tool

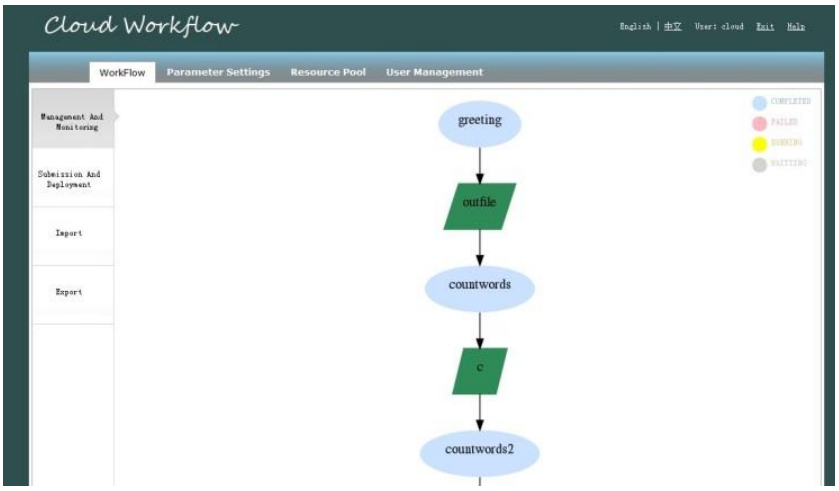


Fig. 6. The Cloud Workflow Management Service

The ***Cloud Resource Manager*** (CRM) accepts resource requests from the Cloud workflow management service, and is in charge of interfacing with OpenNebula and provisioning Falkon virtual clusters dynamically to the workflow service. The process is illustrated in Fig. 8. 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, 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 IO.
3) Upon a resource request from the workflow service:
   a) CRM fetches a VM from the VM pool and starts the Falkon service in that VM.
   b) CRM fetches another VM and starts the Falkon worker in that VM, and also makes that worker register to the Falkon service.
   c) CRM repeats step b) until all the Falkon workers are started and registered.
   d) 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 goes down. If the restart fails, then for a Falkon service VM, it will get a new VM and start Falkon service on it, and have all the workers register to the new service. For a Falkon worker VM, it will replace the worker, and also delete the failed VM.
6) 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.
7) 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 checks the number of the Falkon workers already in the cluster.
   a) If the number is more than requested, then the surplus workers are deregistered and put into the VM pool.
   b) If the number is less than required, then 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. We demonstrate the capability and efficiency of our integration using a small scale experiment setup.

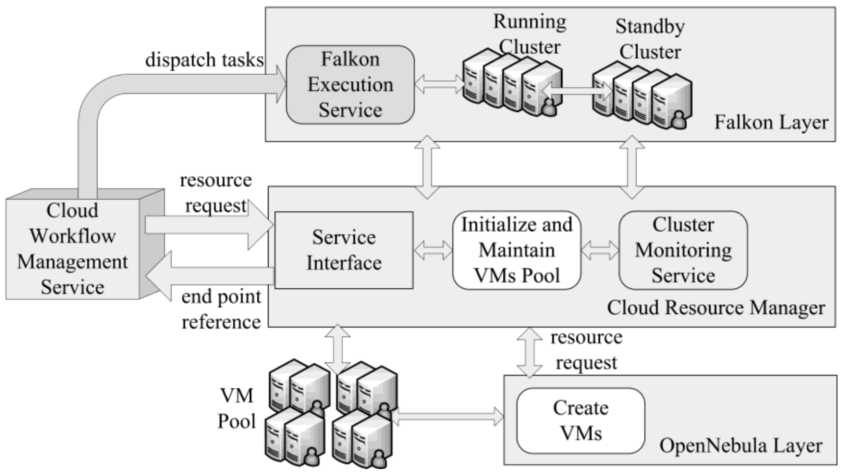


Fig. 7. The Cloud Resource Manager

4 Performance Evaluation

In this section, we demonstrate and analyze our integration approach using a NASA MODIS image processing workflow. The NASA MODIS dataset [30] we use is a set of satellite aerial data blocks, each block is of size around 5.5MB, with digits indicating the geological feature of each point in that block, such as water, sand, green land, urban area, etc.

4.1 The MODIS Image Processing Workflow

The workflow (illustrated in Fig. 9) takes a set of such blocks, gets the size of the urban area in each of the blocks, analyzes and picks the top 12 of the blocks that have the largest urban area, converts them into displayable format, and assembles them into a single PNG file.

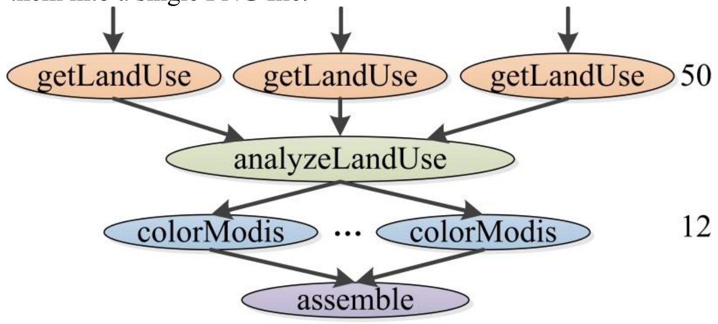


Fig. 8. MODIS Image Processing Workflow

4.2 Experiment Configuration

We use a small cluster setting for the experiments, which includes 6 machines, each configured with Intel Core i5 760 with 4 cores at 2.8GHZ, 4GB memory, 500GB HDD, and connected with Gigabit Ethernet LAN. The operating system is Ubuntu 10.04.1, with OpenNebula 2.2 installed. The configuration for each VM is 1 core, 1.5GB memory, 20GB HDD, and we use KVM as the hypervisor. One of the machines is used as the frontend which hosts the workflow service, the CRM, and the monitoring service. The other 5 machines are used to instantiate VMs, and each physical machine can host up to 2 VMs, so at most 10 VMs can be instantiated in the environment. The configuration of the experiment is illustrated in Fig. 10. Although the cluster size is not significant, we believe it demonstrates the essence of our cluster recycling mechanism.

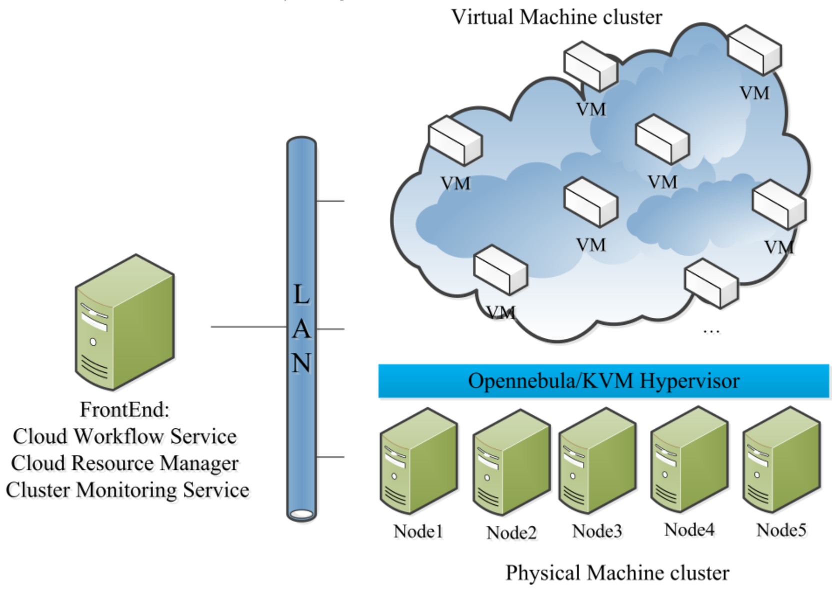


Fig. 9. Experiment Configuration

4.3 Experiment Results

In our experiment, we control the workload by changing the number of input data blocks, the resource required, and the submission type (serial submission or parallel submission). So there are three dependent variables. We design the experiment by making two of the dependent variables constant, and changing the other. We run three types of experiments:

1. The serial submission experiment
2. The parallel submission experiment
3. The different number of data blocks experiment

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

4.3.1 The serial submission experiment

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

In Fig. 11, we can observe that the server creation time is quite stable, around 4.7s every time. Worker creation time is also stable, around 0.6s each, and for worker registration, the first one takes about 10s, and for the rest, about 1s each.

For the rest of the serial submission, we submit a workflow after the previous one has finished to test virtual cluster recycling, where the input data blocks remain fixed.

In Fig. 12, the resources required are one Falkon server with 5 workers, one server with 3 workers and one server with 1 worker. We can see that for the second and third submissions, the worker creation and server creation time are zero, only the surplus workers need to de-register themselves.

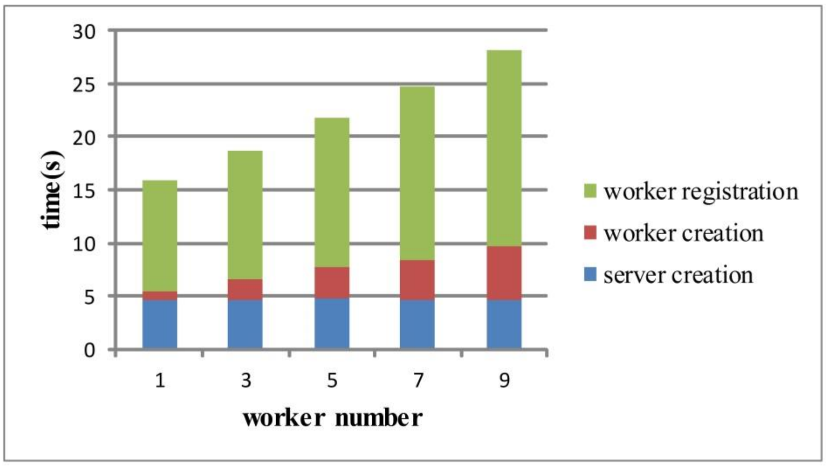


Fig. 10. The Base Line for Cluster Creation

In Fig. 13, the resources required are in the reverse order of those in Fig. 12. Each time two extra Falkon workers need to be created and registered, and the time taken are roughly the same. These experiments show that the Falkon virtual cluster can be re-used after it is being created, and worker resources can be dynamically removed or added.

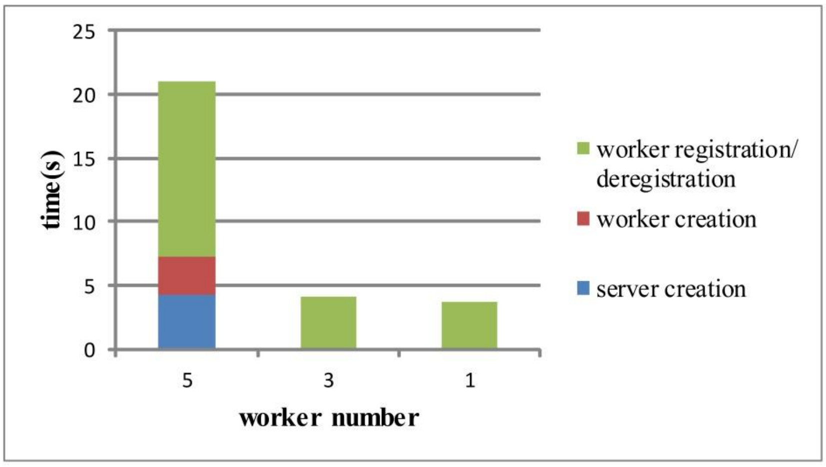


Fig. 11. Serial Submission, Decreasing Resource Required

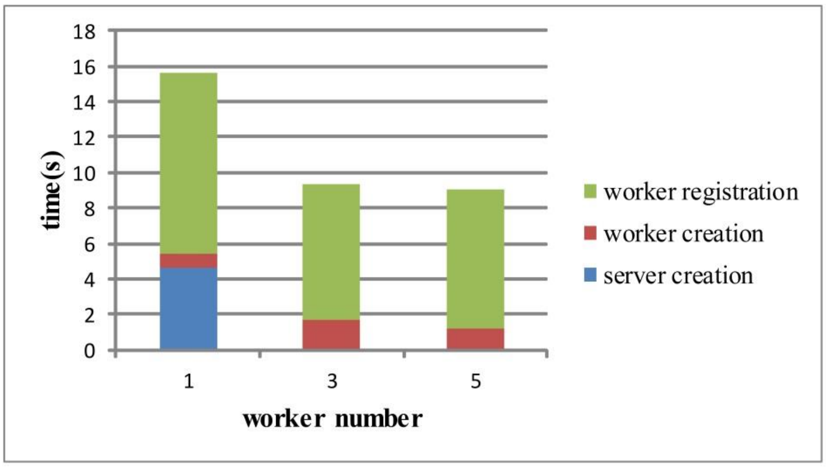


Fig. 12. Serial Submission, Increasing Resource Required

In Fig. 14, we first request a virtual cluster with 1 server and 9 workers, we then make 5 parallel requests for virtual clusters with 1 server and 1 worker. We can observe that one of these requests is satisfied using the existing virtual cluster, where the other 4 are created on-demand. In this case, it takes some time to de-register all the 8 surplus workers, which makes the total time comparable to on-demand creation of the cluster.

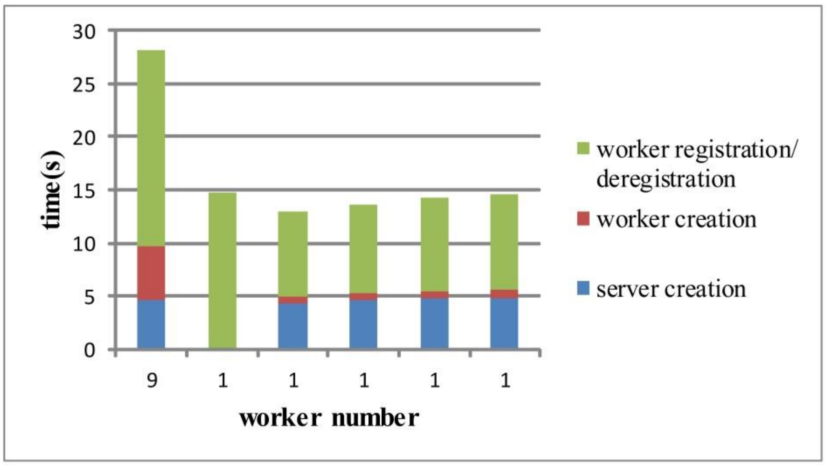


Fig. 13. Serial Submission, Mixed Resource Required

4.3.2 The parallel submission experiment

In the parallel submission experiment, we submit multiple workflows at the same time in order to measure the maximum parallelism (the number of concurrent workflows that can be hosted in the Cloud platform) in the environment.

First, we submit resource requests with 1 server and 2 workers, and the maximum parallelism is up to three. In Table 1, we show the results for the experiment, in which we make resource requests for 1 virtual cluster, 2 virtual clusters, 3 virtual clusters and 4 virtual clusters.

TABLE 1 Parallel Submission, 1 Server and 2 Workers

# of Clusters	Server Creation	Worker Creation	Worker Registration
1	4624ms	1584ms	11305ms
2	4696ms	2367ms	11227ms
	445ms	0	0
3	4454ms	1457ms	11329ms
	488ms	0	0
	548ms	0	0
4	521ms	0	0
	585ms	0	0
	686ms	0	0
	submission failed		

For the request of 2 virtual clusters, it can re-use the one released by the early request, and the time to initialize the cluster is significantly less than fresh creation (445ms vs. 4696ms). It has to create the second cluster on-demand. For the 4-virtual-cluster request, since all the VM resources are used up by the first 3 clusters, the 4th cluster creation would fail as expected. When we change resource requests to 1 server and 4 workers, the maximum parallelism is two, and the request to create a third virtual cluster also fails. Since our VM pool has a maximum of ten virtual machines, it's easy to explain why this has happened. This experiment shows that our integrated system can maximize the cluster resources assigned to workflows to achieve efficient utilization of resources.

4.3.3 Different number of data blocks experiment

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. 15, 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 can’t subsidize the management and registration overhead of the added worker. The time for server and worker creation, and worker registration remain unchanged when we change the input size (as have been shown in Fig. 11). 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.

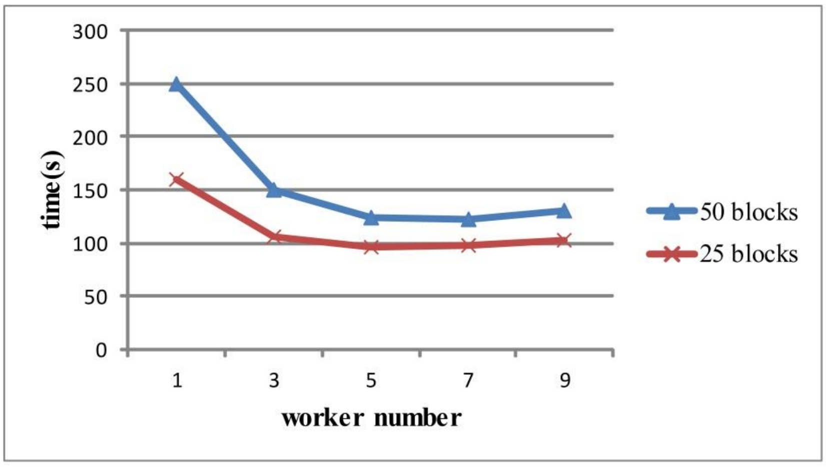


Fig. 14. Different Input Sizes

5 Related Work

Systems such as Taverna [11], Kepler [9], Vistrails [10], Pegasus [8], Swift [32], and VIEW [26] have seen wide adoption in various disciplines such as Physics, Astronomy, Bioinformatics, Neuroscience, Earth Science, and Social Science.

In Table 2, we list some use cases that focused on applying SWFMSs to execute data-intensive applications.

TABLE 2 Use Cases of SWFMSs

SWFMSs	Application Fields	Use Cases
Swift	Climate Science	Climate Data Analysis[13]
Taverna	Bioinformatics	Single Nucleotide Polymorphisms Analysis[14]
Vistrails	Earth Science	NASA Earth Exchange [15]
Kepler	Physics	Hyperspectral image processing [37]
VIEW	Medical Science	Neurological disorder diagnosis[48]

There are some early explorers that try to evaluate the feasibility, performance, and adaptation of running data intensive and HPC applications on Clouds or hybrid Grid/Cloud environments. Palankar et al. [17] 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. [35] evaluated the performance of X-Ray Crystallography workflow using SciCumulus middleware with Amazon EC2. Wang et al. [36] 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 source of information about Cloud platform support for science applications. Other studies investigated the execution of real science applications on commercial Clouds [19] [20], 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 doesn't show significant benefit due to the applications' tightly coupled nature.

There have been a couple of researcher that have been investigating techniques for deploying data-intensive workflows in the cloud using unique architectures that are difficult to deploy on traditional platforms, such as grids [55–57]. Meanwhile, some other researches focused on developing new algorithms for workflows to take advantage of the unique pricing model and elasticity of infrastructure clouds [49-53], and investigating new cloud workflow-scheduling algorithms that optimize for cost, performance, and other quality-of-service metrics [58 –60].

Gideon Juve et al. have studied the cost and performance of workflows in the cloud via simulation [24], using an experimental Nimbus cloud [25], individual Elastic Compute Cloud (EC2) nodes [28], and a variety of different intermediate storage systems on EC2 [43]. Christian Vecchiola et al. have done similar investigations on EC2 and Grid5000 [54]. 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, Gideon Juve et al [41] also related the practical experience of trying to run a nontrivial scientific workflow application on three different infrastructures and compare the benefits and challenges of each platform.

M. Kozlovszky et al. [42] introduced a convenient way for sharing, integrating and executing different workflows in heterogeneous infrastructure environments. The paper explained in detail how to enable generic DCI compatibility for grid workflow management systems (such as ASKALON, MOTEUR, gUSE/WS-PGRADE, etc.) on job level and indirectly on workflow level. The generic DCI Bridge service enables the execution of jobs onto existing major DCI platforms (such as Service Grids, Desktop Grids, Web services, or even Cloud based DCIs). The CODA framework [16] was designed and implemented to support big data analytics in Cloud computing. Important functions, such as workflow scheduling, data locality, resource provisioning, and monitoring functions, had been integrated into the framework. Through the CODA framework, the workflows could be easily composed and efficiently executed in Amazon EC2. In order to address performance and cost issues of big data processing on Clouds, Long Wang et al. [5] presented a novel design of adaptive workflow management system which included a data mining based prediction model, workflow scheduler, and iteration controls to optimize the data processing via iterative workflow tasks.

Those works mentioned above are important as they provide valuable experience on migrating traditional scientific workflows to various Cloud platforms. However, a normalized, end-to-end integration approach is still missing. We present an end-to-end approach that addresses the integration of Swift, an SWFMS that has broad application in Grids and supercomputers, with the OpenNebula Cloud platform. The integration covers all the major aspects involved in workflow management in the Cloud, including a client side workflow submission tool, a Cloud workflow management service, a Cloud Resource Manager (CRM), and a cluster monitoring service.

6 Conclusion and Future Work

As more and more scientific applications are migrating into Cloud, it is imperative to also migrate SWFMSs into Cloud to take advantage of Cloud scalability, and also to handle the ever increasing data scale and analysis complexity of such applications. Cloud offers unprecedented scalability to workflow systems, and could potentially change the way we perceive and conduct scientific experiments. The scale and complexity of the science problems that can be handled can be greatly increased on the Cloud, and the on-demand nature of resource allocation on the Cloud will also help improve resource utilization and user experience.

We first introduce our service framework for integrating SWFMSs into Cloud computing platforms. Then we present our early effort in integrating the Swift workflow management system with the OpenNebula Cloud platform, in which a Cloud workflow management service, a Cloud resource manager, and a cluster monitoring service are developed. We also demonstrate the functionality and efficiency of our approach using a set of experiments and a real-world scientific workflow.

For future work, we are working on a common interface that will facilitate the integration of Swift with other Cloud platforms such as Amazon EC2 and Openstack. We will also investigate commonality in migrating other SWFMSs into Cloud, i.e. ways to offer SWFMSs as a service and to enable them to interact with the underlying Cloud resources. We will also leverage distributed storage for VM images for more efficient access, and conduct large scale experiments to look at ways to improve VM instantiation, virtual cluster creation and workflow execution. We are also exploring redesigning workflow systems from the ground up using Cloud Computing building blocks, such as EC2 [68], SQS [66], DynamoDB [67], S3 [65], and CloudWatch [68], in order to deliver a light-weight, fast, and distributed workflow system that should scale along with the largest cloud infrastructures [44].

Acknowledgments

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

7 References

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

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

[3] GenBank, [Online]. Available: http://www.ncbi.nlm.nih.gov/genbank, 2014

[4] Large Hadron Collider, [Online]. Available: http://lhc.web.cern.ch, 2014

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

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

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

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

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

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

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

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

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

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

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

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

[17] 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), pp. 55-64, 2008.

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

[19] E. Deelman, G. Singh, M. Livny, B. Berriman, and 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, pp. 50:1–50:12, Piscataway, NJ, USA, 2008.

[20] C. Vecchiola, S. Pandey, and R. Buyya. High-Performance Cloud Computing: A View of Scientific Applications. In International Symposium onParallel Architectures, Algorithms, and Networks, pp. 4-16, 2009.

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

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

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

[24] Deelman E, Singh G, Livny M, et al. The cost of doing science on the cloud: the montage example[C]//Proceedings of the 2008 ACM/IEEE conference on Supercomputing. IEEE Press, 2008: 50.

[25] Hoffa C, Mehta G, Freeman T, et al. On the use of cloud computing for scientific workflows[C]//eScience, 2008. eScience'08. IEEE Fourth International Conference on. IEEE, 2008: 640-645.

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

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

[28] Juve G, Deelman E, Vahi K, et al. Scientific workflow applications on Amazon EC2[C]//E-Science Workshops, 2009 5th IEEE International Conference on. IEEE, 2009: 59-66.

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

[30] NASA MODIS dataset, [Online]. Available: http://modis.gsfc.nasa. gov/, 2013.

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

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

[33] Hadoop, [Online]. Available: http://hadoop.apache.org/, 2012

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

[35] Oliveira, D. Ocaña, K., Ogasawara, E., Dias, J., Baião, F., Mattoso, M., A Performance Evaluation of X-Ray Crystallography Scientific Workflow Using SciCumulus. IEEE CLOUD 2011, pp. 708-715.

[36] L. Wang, J. Tao, M. Kunze, A. C. Castellanos, D. Kramer, and W. Karl, "Scientific Cloud Computing: Early Definition and Experience," in 10th IEEE International Conference on High Performance Computing and Communications, HPCC '08. , pp. 825-830, 2008.

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

[38] R. Moreno-Vozmediano, R.S. Montero, I.M. Llorente. *"Multi-Cloud Deployment of Computing Clusters for Loosely-Coupled MTC Applications",* IEEE Transactions on Parallel and Distributed Systems. 22(6), pp.924-930, 2011.

[39] R. S. Montero, R. Moreno-Vozmediano, and I. M. Llorente. *"An Elasticity Model for High Throughput Computing Clusters",* J. Parallel and Distributed Computing. 71(6), pp.750-757, 2011.

[40] OpenNebula Architecture, http://www.opennebula.org/documentation:archives:rel2.2:architecture, 2013.

[41] Juve G, Rynge M, Deelman E, et al. Comparing FutureGrid, Amazon EC2, and Open Science Grid for Scientific Workflows[J]. Computing in Science & Engineering, 2013, 15(4): 20-29.

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

[43] Juve G, Deelman E, Vahi K, et al. Data sharing options for scientific workflows on amazon ec2[C]//Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE Computer Society, 2010: 1-9.

[44] I. Sadooghi, I. Raicu. "CloudKon: a Cloud enabled Distributed tasK executiON framework", Illinois Institute of Technology, Department of Computer Science, PhD Oral Qualifier, 2013

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

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

[47] Service Framework, [Online]. Available: http://www.cloud-uestc.cn/projects/serviceframework/index.html

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

[49] Lin C, Lu S. Scheduling scientific workflows elastically for cloud computing[C]//Cloud Computing (CLOUD), 2011 IEEE International Conference on. IEEE, 2011: 746-747.

[50] Mao M, Humphrey M. Auto-scaling to minimize cost and meet application deadlines in cloud workflows[C]//Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. ACM, 2011: 49.

[51] Oliveira D, Ogasawara E, Ocaña K, et al. An adaptive parallel execution strategy for cloud‐based scientific workflows[J]. Concurrency and Computation: Practice and Experience, 2012, 24(13): 1531-1550.

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

[53] Reynolds C J, Winter S, Terstyanszky G Z, et al. Scientific workflow makespan reduction through cloud augmented desktop grids[C]//Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on. IEEE, 2011: 18-23.

[54] Vecchiola C, Pandey S, Buyya R. High-performance cloud computing: A view of scientific applications[C]//Pervasive Systems, Algorithms, and Networks (ISPAN), 2009 10th International Symposium on. IEEE, 2009: 4-16.

[55] Yuan D, Yang Y, Liu X, et al. On-demand minimum cost benchmarking for intermediate dataset storage in scientific cloud workflow systems[J]. Journal of Parallel and Distributed Computing, 2011, 71(2): 316-332.

[56] Çatalyürek Ü V, Kaya K, Uçar B. Integrated data placement and task assignment for scientific workflows in clouds[C]//Proceedings of the fourth international workshop on Data-intensive distributed computing. ACM, 2011: 45-54.

[57] Wang J, Korambath P, Altintas I. A physical and virtual compute cluster resource load balancing approach to data-parallel scientific workflow scheduling[C]//Services (SERVICES), 2011 IEEE World Congress on. IEEE, 2011: 212-215.

[58] Tolosana-Calasanz R, BañAres J Á N, Pham C, et al. Enforcing QoS in scientific workflow systems enacted over Cloud infrastructures[J]. Journal of Computer and System Sciences, 2012, 78(5): 1300-1315.

[59] Bessai K, Youcef S, Oulamara A, et al. Bi-criteria workflow tasks allocation and scheduling in Cloud computing environments[C]//Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on. IEEE, 2012: 638-645.

[60] Ostermann S, Prodan R. Impact of variable priced cloud resources on scientific workflow scheduling[M]//Euro-Par 2012 Parallel Processing. Springer Berlin Heidelberg, 2012: 350-362.

[61] Ioan Raicu. "Many-Task Computing: Bridging the Gap between High Throughput Computing and High Performance Computing", Computer Science Department, University of Chicago, Doctorate Dissertation, March 2009

[62] Ioan Raicu, Ian Foster, Yong Zhao, Alex Szalay, Philip Little, Christopher M. Moretti, Amitabh Chaudhary, Douglas Thain. "Towards Data Intensive Many-Task Computing", book chapter in "Data Intensive Distributed Computing: Challenges and Solutions for Large-Scale Information Management", IGI Global Publishers, 2009

[63] Michael Wilde, Ioan Raicu, Allan Espinosa, Zhao Zhang, Ben Clifford, Mihael Hategan, Kamil Iskra, Pete Beckman, Ian Foster. "Extreme-scale scripting: Opportunities for large task-parallel applications on petascale computers", Scientific Discovery through Advanced Computing Conference (SciDAC09) 2009

[64] Dongfang Zhao, Chen Shou, Tanu Malik, Ioan Raicu. “Distributed Data Provenance for Large-Scale Data-Intensive Computing”, IEEE Cluster 2013

[65] Ioan Raicu, Pete Beckman, Ian Foster. “Making a Case for Distributed File Systems at Exascale”, ACM Workshop on Large-scale System and Application Performance (LSAP), 2011

[66] Dharmit Patel, Faraj Khasib, Iman Sadooghi, Ioan Raicu. "Towards In-Order and Exactly-Once Delivery using Hierarchical Distributed Message Queues", 1st International Workshop on Scalable Computing For Real-Time Big Data Applications (SCRAMBL'14) 2014

[67] Tonglin Li, Xiaobing Zhou, Kevin Brandstatter, Dongfang Zhao, Ke Wang, Anupam Rajendran, Zhao Zhang, Ioan Raicu. “ZHT: A Light-weight Reliable Persistent Dynamic Scalable Zero-hop Distributed Hash Table”, IEEE International Parallel & Distributed Processing Symposium (IPDPS) 2013

[68] Iman Sadooghi, Sandeep Palur, Ajay Anthony, Isha Kapur, Karthik Belagodu, Pankaj Purandare, Kiran Ramamurty, Ke Wang, Ioan Raicu. "Achieving Efficient Distributed Scheduling with Message Queues in the Cloud for Many-Task Computing and High-Performance Computing", 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid) 2014