Using Cloud-Aware Provenance to Reproduce Scientific Workflow Execution on Cloud

Khawar Hasham, Kamran Munir and Richard McClatchey
Centre for Complex Computing Systems (CCCS), Faculty of Environment and Technology (FET), University of the West of England (UWE), Bristol, U.K.

Keywords: Cloud Computing, Scientific Workflows, Provenance, Reproducibility, Repeatability.

Abstract: Provenance has been thought of a mechanism to verify a workflow and to provide workflow reproducibility. This provenance of scientific workflows has been effectively carried out in Grid based scientific workflow systems. However, recent adoption of Cloud-based scientific workflows present an opportunity to investigate the suitability of existing approaches or propose new approaches to collect provenance information from the Cloud and to utilize it for workflow reproducibility on the Cloud infrastructure. This paper presents a novel approach that can assist in mitigating this challenge. This approach can collect Cloud infrastructure information along with workflow provenance and can establish a mapping between them to provide a Cloud-aware provenance. The reproducibility of the workflow execution is performed by: (a) capturing the Cloud infrastructure information (virtual machine configuration) along with the workflow provenance, (b) re-provisioning the similar resources on the Cloud and re-executing the workflow on them and (c) by comparing the outputs of workflows. The evaluation of the prototype suggests that the proposed approach is feasible and can be investigated further. Since there is no reference model for workflow reproducibility on Cloud exists in the literature, this paper also attempts to present a model that is used in the proposed design to achieve workflow reproducibility in the Cloud environment.

1 INTRODUCTION

The scientific community is processing and analysing huge amounts of data being generated in modern scientific experiments that include projects such as DNA analysis (Foster et al., 2008), the Large Hadron Collider (LHC) (http://lhc.cern.ch), and projects such as neuGRID (Mehmood et al., 2009) and its follow-on neuGRIDforUsers (Munir et al., 2013, 2014). In particular the neuGRID community is utilising scientific workflows to orchestrate the complex analysis of neuro-images to diagnose Alzheimer disease. A large pool of compute and data resources are required to process this data, which has been available through the Grid (Foster et al., 1999) and is now also being offered by the Cloud-based infrastructures.

Cloud computing (Mell and Grance, 2011) has emerged as a new computing and storage paradigm, which is dynamically scalable and usually works on a pay-as-you-go cost model. It aims to share resources to store data and to host services transparently among users at a massive scale (Mei et al., 2008). Its ability to provide an on-demand computing infrastructure enables distributed processing of scientific workflows (Deelman et al., 2008) with increased complexity and data requirements. Recent work (Juve and Deelman 2010) is now experimenting with Cloud infrastructures to assess the feasibility of executing workflows on the Cloud.

An important consideration during this data processing is to gather provenance (Simmhan et al., 2005) information that can provide detailed information about both the input and the processed output data, and the processes involved in a workflow execution. This information can be used to debug the execution of a workflow, to aid in error tracking and reproducibility. This vital information can enable scientists to verify the outputs and iterate on the scientific method, to evaluate the process and results of other experiments and to share their own experiments with other scientists (Azarnoosh et al., 2013). The execution of scientific workflows in the Cloud brings to the fore the need to collect provenance information that is necessary to ensure the reproducibility of these experiments on the Cloud infrastructure

A research study (Belhajjame et al., 2012) conducted to evaluate the reproducibility of scientific workflows has shown that around 80% of the workflows cannot be reproduced, and 12% of them are due to the lack of information about the execution environment. This information affects a workflow on two levels. It can affect a workflow's overall execution performance and also job failure rate. For instance, a data-intensive job can perform better with 2GB of RAM because it can accommodate more data in RAM, which is a faster medium than hard disk. However, the job's performance will degrade if a resource of 1GB RAM is allocated to this job as less data can be placed in RAM. Moreover, it is also possible that jobs will remain in waiting queues or fail during execution if their required hardware dependencies are not met. This becomes a more challenging issue in the context of Cloud in which resources can be created or destroyed at runtime.

The dynamic and geographically distributed nature of Cloud computing makes the capturing and processing of provenance information a major research challenge (Vouk 2008, Zhao et al., 2011). Since the Cloud presents a transparent access to dynamic execution resources, the workflow parameters including execution resource configuration should also be known to a scientist (Shamdasani et al., 2012) i.e. what execution environment was used for a job in order to reproduce a workflow execution on the Cloud. Due to these reasons, there is a need to capture information about the Cloud infrastructure along with workflow provenance, to aid in the reproducibility of workflow experiments. There has been a lot of research related to provenance in the Grid (Foster et al., 2002, Stevens et al., 2003) and a few initiatives (Oliveira et al., 2010, Ko et al., 2011) for the Cloud. However, they lack the information that can be utilised for re-provisioning of resources on the Cloud, thus they cannot create the similar execution environment(s) for workflow reproducibility. In this paper, the terms "Cloud infrastructure" and "virtualization layer" are used interchangeably.

This paper presents a theoretical description of an approach that can augment workflow provenance with infrastructure level information of the Cloud and use it to provision similar execution environment(s) and repeat a given workflow. Important points discussed in this paper are as follows: section 2 presents some related work in provenance related systems. Section 3 presents a reproducibility model designed after collecting guidelines used and discussed in literature. Section 4 presents an overview of the proposed approach. Section 5 presents an evaluation of the developed prototype. And finally section 6 presents some conclusions and directions for future work.

2 RELATED WORK

Significant research (Foster et al., 2002, Scheidegger et al., 2008) has been carried out in workflow provenance for Grid-based workflow management systems. Chimera (Foster et al., 2002) is designed to manage the data-intensive analysis for high-energy physics (GriPhyN) (GriPhyN 2014) and astronomy (SDSS) (SDSS 2014) communities. It captures process information, which includes the runtime parameters, input data and the produced data. It stores this provenance information in its schema, which is based on a relational database. Although the schema allows storing the physical location of a machine, it does not support the hardware configuration and software environment in which a job was executed. Vistrails (Scheidegger et al., 2008) provides support for scientific data exploration and visualization. It not only captures the execution log of a workflow but also the changes a user makes to refine his workflow. However, it does not support the Cloud virtualization layer information. Similar is the case with Pegasus/Wings (Kim et al. 2008) that supports evolution of a workflow. However, this paper is focusing on the workflow execution provenance on the Cloud, rather than the provenance of a workflow itself (e.g. design changes).

There have been a few research studies (Oliveira et al., 2010, Ko et al., 2011) performed to capture provenance in the Cloud. However, they lack the support for workflow reproducibility. Some of the work in Cloud towards provenance is directed to the file system (Zhang et al., 2011, Shyang et al 2012) or hypervisor level (Macko et al., 2011). However such work is not relatable to our approach because this paper focuses on virtualized layer information of the Cloud for workflow execution. Moreover, the collected provenance data provides information about the file access but it does not provide information about the resource configuration. The PRECIP (Azarnoosh et al., 2013) project provides an API to provision and execute workflows. However, it does not provide provenance information of a workflow.

There have been a few recent projects (Chirigati et al., 2013, Janin et al., 2014) and research studies (Perez et al., 2014a) on collecting provenance and using it to reproduce an experiment. A semantic-based approach (Perez et al., 2014b) has been proposed to improve reproducibility of workflows in the Cloud. This approach uses ontologies to extract information about the computational environment from the annotations provided by a user. This information is then used to recreate (install or configure) that environment to reproduce a workflow execution. On the contrary, our approach is not relying on annotations rather it directly interacts with the Cloud middleware at runtime to acquire resource configuration information and then establishes mapping between workflow jobs and Cloud resources. The ReproZip software (Chirigati et al., 2013) uses system call traces to provide provenance information for job reproducibility and portability. It can capture and organize files/libraries used by a job. The collected information along with all the used system files are zipped together for portability and reproducibility purposes. Since this approach is useful at individual job level, this does not work for an entire workflow, which is the focus of this paper. Moreover, this approach does not consider the hardware configuration of the underlined execution machine. Similarly, a Linux-based tool, CARE (Janin et al., 2014), is designed to reproduce a job execution. It builds an archive that contains selected executable/binaries and files accessed by a given job during an observation run.

3 WORKFLOW REPRODUCIBILITY MODEL ON CLOUD

As per our understanding of the literature, there is not a standard reproducibility model proposed so far for scientific workflows, especially in Cloud environment. However, there are some guidelines or policies, which have been highlighted in literature to reproduce experiments. There is one good effort (Sandve et al., 2013) in this regard, but it mainly talks about reproducible papers and it does not consider execution environment of workflows. In this section, we have gathered basic points to present an initial workflow reproducibility model in Cloud that can provide guidelines for future work in this regard. These points are discussed as follows.

- Share Code and Data

The need for data and code sharing in computational science has been widely discussed (Stodden 2010). In computational science conservation, in particular for scientific workflow executions, it is emphasized that the data, code, and the workflow description should be available in order to reproduce an experiment.

- Execution Infrastructure details

The execution infrastructure provided by the Grid or Cloud to execute a workflow is composed of a set of computational and storage resources (e.g. execution nodes, storage devices, networking). The physical approach, where actual computational hardware are made available for long time period to scientists, often conserves the computational environment including supercomputers, clusters, or Grids (Perez et al., 2014b). As a result, scientists are able to reproduce their experiments on the same hardware environment. However, this luxury is not available in the Cloud environment in which resources are virtualized and provisioned dynamically on-demand. A little focus is given to the underlying infrastructure, especially Cloud, in computational conservation in literature. This challenge has been tackled in this paper by collecting this information at runtime from the Cloud infrastructure. From resource provisioning point of view, parameters such as RAM, vCPU and Hard Disk are important in selecting appropriate resource especially on the Cloud and should be made part of the collected provenance. All these factors contribute to the job's execution performance as well as to its failure rate. For instance, a job will fail if it is scheduled to a resource with insufficient available RAM.

- Software Environment

Apart from knowing the hardware infrastructure, it is also essential to provide information about the software environment. A software environment determines the operating system and the libraries used to execute a job. Without the access to required libraries information, a job execution will fail. For example, a job, relying on MATLAB library, will fail in case the required library is missing. One possible approach (Howe et al., 2012) to conserve software environment is thought to conserve VM that is used to execute a job and then reuse the same VM while re-executing the same job. One may argue that it would be easier to keep and share VM images with the research community through a common repository, however the high storage demand of VM images remains a challenging problem (Zhao et al., 2014). In the prototype presented in this paper, the OS image used to provision a VM is conserved and thought to present all the software dependencies required for a job execution in a workflow. Therefore, the proposed solution should also retrieve the image information to build a virtual machine on which the workflow job was executed.

- Workflow Versioning

Capturing only a provenance trace is not sufficient to allow a computation to be repeated – a situation known as workflow decay (Roure et al., 2011). The reason is that the provenance systems can store information on how the data was generated, however they do not store copies of the key actors in the computation i.e. workflow, services, data. This paper (Sandve et al. 2013) suggests to archive the exact versions of all programs and enable version control on all scripts used in an experiment. This is not supported in the presented prototype, but it will be incorporated in next iterations.

- Provenance Comparison

The provenance traces of two executed workflows should be compared to determine workflow reproducibility. The main idea is to evaluate the reproducibility of an entire execution of a given workflow, including the logical chaining of activities and the data. To provide the strict reproducibility functionality, a system must guarantee that the data are still accessible and that the corresponding activities are accessible (Lifschitz et al. 2011). Since the focus of this paper is on workflow reproducibility on the Cloud infrastructure, the execution infrastructure should also be part of the comparison. Therefore the provenance comparison should be made at different levels; workflow structure, execution infrastructure, and workflow input and output. A brief description of this comparison is given below.

a) Workflow structure should be compared to determine that both workflows are similar. Because it is possible that two workflows are having similar number of jobs but with different job execution order.
b) Execution infrastructure (software environment, resource configuration) used on the Cloud for a workflow execution should also be compared.
c) Comparison of input and output should be made to evaluate workflow reproducibility. There could be a scenario that a user repeated a workflow but with different inputs, thus producing different outputs. It is also possible that changes in job or software library result into different workflow output. There are a few approaches (Missier et al. 2013), which perform workflow provenance comparison to determine differences in reproduced workflows. The proposed approach in this paper incorporates the workflow output comparison to determine the reproducibility of a workflow.

- Pricing Model

This point can be important for experiments in which cost is also a main factor. The resource provisioned on the Cloud has associated cost, which is based on the resource type and the amount of time it has been used for. This information can assist in reproducing an experiment with the same cost as was incurred in earlier execution. This point is not incorporated in the proposed design at the moment.

4 CLOUD-AWARE PROVENANCE APPROACH

An abstract view of the proposed architecture is presented in this section. This architecture is designed after evaluating the existing literature and keeping in mind the objectives of this research study. The proposed architecture is inspired by the mechanism used in a paper (Groth et al., 2009) for executing workflows on the Cloud. Figure 1 illustrates the proposed architecture that is used to capture the Cloud infrastructure information and to interlink it with the workflow provenance collected from a workflow management system such as Pegasus. This augmented or extended provenance information compromising of workflow provenance and the Cloud infrastructure information is named as Cloud-aware provenance. The components of this architecture are briefly explained below.

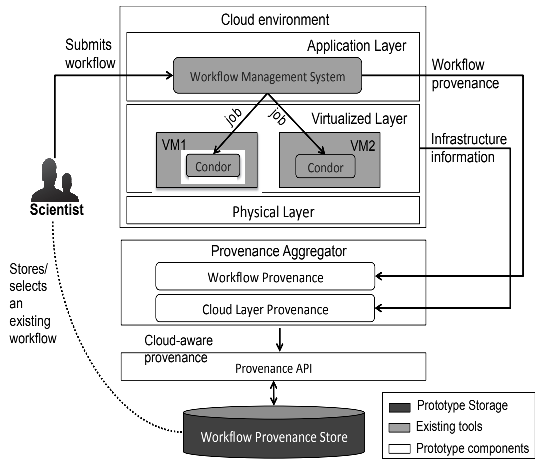

Figure 1: An abstract architecture of the proposed approach.

- **Workflow Provenance:** This component is responsible for receiving provenance captured at the application level by the workflow management system (Pegasus). Since workflow management systems may vary, a plugin-based approach is used for this component. Common interfaces are designed to develop plugins for different workflow management systems. The plugin also translates the workflow provenance according to the representation that is used to interlink the workflow provenance along with the information coming from the Cloud infrastructure.

- **Cloud Layer Provenance:** This component is responsible for capturing information collected from different layers of the Cloud. To achieve re-provisioning of resources on Cloud, this component focuses on the virtualization layer and retrieves information related to the Cloud infrastructure i.e. virtual machine configuration. This component is discussed in detail in section 4.1.
- **Provenance Aggregator:** This is the main component tasked to collect and interlink the provenance coming from different layers as shown in Figure 1. It establishes interlinking connections between the workflow provenance and the Cloud infrastructure information. The provenance information is then represented in a single format that could be stored in the provenance store through the interfaces exposed by the Provenance API.
- **Provenance API:** This acts as a thin layer to expose the provenance storage capabilities to other components. Through its exposed interfaces, outside entities such as the Provenance Aggregator would interact with it to store the workflow provenance information. This approach gives flexibility to implement authentication or authorization in accessing the provenance store.
- **Workflow Provenance Store:** This data store is designed to store workflows and their associated provenance. This also keeps mapping between workflow jobs and the virtual compute resources in the Cloud infrastructure. This also keeps record of the workflow and its related configuration files being used to submit a user analysis on the Cloud. This information is later retrieved to reproduce the execution. However, it does not support workflow evolution in its current design.

4.1 Job to Cloud Resource Mapping

The CloudLayerProvenance component is designed in a way that interacts with the Cloud infrastructure as an outside client to obtain the resource configuration information. As mentioned earlier, this information is later used for reprovisioning the resources to provide a similar execution infrastructure to repeat a workflow execution. Once a workflow is executed, Pegasus collects the provenance and stores it in its own internal database. Pegasus also stores the IP address of the virtual machine (VM) where the job is executed. However, it lacks other VM specifications such as RAM, CPUs, hard disk etc. The CloudLayerProvenance component retrieves all the jobs of a workflow and their associated VM IP addresses from the Pegasus database. It then collects a list of virtual machines owned by a respective user from the Cloud middleware. Using the IP address, it establishes a mapping between the job and the resource configuration of the virtual machine used to execute the job. This information i.e. Cloud-aware provenance is then stored in the Provenance Store. The flowchart of this mechanism is presented in Figure 2.

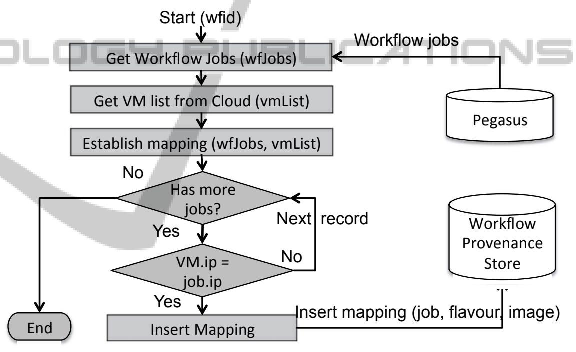
Start (wfid)
Workflow jobs
Get Workflow Jobs (wfJobs)
Get VM list from Cloud (vmList)
Pegasus
Establish mapping (wfJobs, vmList)
No
Has more jobs?
Next record
Yes
Workflow Provenance Store
VM.ip = job.ip
No
Yes
End
Insert Mapping
Insert mapping (job, flavour, image)

Figure 2: flowchart of job to Cloud resource mapping performed by ProvenanceAggregator component.

In this flowchart, the variable *wfJobs* – representing a list of jobs of a given workflow – is retrieved from the Pegasus database. The variable *vmList* – represents a list of virtual machines in the Cloud infrastructure – is collected from the Cloud. A mapping between jobs and VMs is established by matching the IP addresses (see in Figure 2). Resource configuration parameters such as flavour and image are obtained once the mapping is established. *flavour* defines resource configuration such as RAM, Hard disk and CPUs, and *image* defines the operating system image used in that particular resource. By combining these two parameters together, one can provision a resource on the Cloud infrastructure. After retrieving these parameters and jobs, the mapping information is then stored in the Provenance Store (see in Figure. 2). This mapping information provides two important data (a) hardware configuration (b) software configuration using VM name. As discussed in section 3, these two parameters are important in recreating a similar execution environment.

4.2 Workflow Reproducibility using Cloud-Aware Provenance

In section 4.1, the job to Cloud resource mapping using provenance information has been discussed. This mapping is stored in the database for workflow reproducibility purposes. In order to reproduce a workflow execution, researcher first needs to provide the *wfID* (workflow ID), which is assigned to every workflow in Pegasus, to the proposed framework to re-execute the workflow using the Cloud-aware provenance. It retrieves the given workflow from the Provenance Store database (step 2(a) in Figure 3) along with the Cloud resource mapping stored against this workflow (step 2(b) in Figure 3). Using this mapping information, it retrieves the resource flavour and image configurations, and provisions the resources (step 3 in Figure 3) on Cloud. Once resources are provisioned, it submits the workflow (step 4).

At this stage, a new workflow ID is assigned to this newly submitted workflow. This new wfID is passed over to the ProvenanceAggregator component to monitor (step 5) the execution of the workflow and start collecting its Cloud-aware provenance information (see step 6 in Figure 3) This is important to recollect the provenance of the repeated workflow, as this will enable us to verify the provisioned resources by comparing their resource configurations with the old resource configuration.

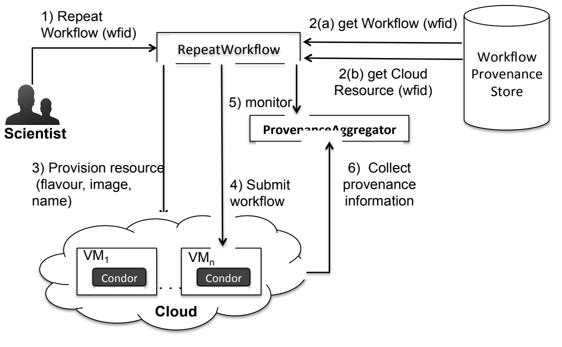
1) Repeat Workflow (wfid)
RepeatWorkflow
2(a) get Workflow (wfid)
Workflow Provenance Store
2(b) get Cloud Resource (wfid)
5) monitor
Scientist
ProvenanceAggregator
3) Provision resource (flavour, image, name)
4) Submit workflow
6) Collect provenance information
VM1
Condor
VMn
Condor
Cloud

Figure 3: The sequence of activities to illustrate workflow repeatability in the proposed system.

4.3 Workflow Output Comparison

Another aspect of workflow repeatability is to verify that it has produced the same output that was produced in its earlier execution (as discussed in section 3). In order to evaluate workflow repeatability, an algorithm has been proposed that compares the outputs produced by two given workflows. It uses the MD5 hashing algorithm (Stalling 2010) on the outputs and compares the hash value to verify the produced outputs. The two main reasons of using a hash function to verify the produced outputs are; a) simple to implement and b) the hash value changes with a single bit change in the file. If the hash values of two given files are same, this means that the given files contain same content.

The proposed algorithm (as shown in Figure 4) operates over the two given workflows identified by *srcWfID* and *destWfID*, and compares their outputs. It first retrieves the list of jobs and their produced output files from the Provenance Store for each given workflow. It then iterates over the files and compares the source file, belonging to *srcWfID*, with the destination file, belonging to *destWfID*. Since the files are stored on the Cloud, the algorithm retrieves the files from the Cloud (see lines 11 and 12). Cloud storage services such as OpenStack Swift, Amazon Object Store use the concept of a bucket or a container to store a file. This is why *src_container* and *dest_container* along with *src_filename* and *dest_filename* are given in the *GetCloudFile* function to identify a specific file in the Cloud. The algorithm then compares the hash value of both files and increments *ComparisonCounter*. If all the files in both workflows are the same, *ComparisonCounter* should be equal to *FileCounter*, which counts the number of files produced by a workflow. Thus, it confirms that the workflows are repeated successfully. Otherwise, the algorithms returns false if both these counters are not equal.

```
Algorithm 3 Compare Workflow Outputs Algorithm
Require: srcWfID : Source Workflow ID.
         destWfID : Destination Workflow ID

 1: procedure COMPAREWORKFLOWOUTPUTS(srcWfID, destWfID)
 2:     srcWorkflowJobs ← GETWORKFLOWJOBS(srcWfID)
 3:     destWorkflowJobs ← GETWORKFLOWJOBS(destWfID)
 4:     FileCounter ← 0
 5:     ComparisonCounter ← 0
 6:     for all jobfiles ∈ srcWorkflowJobs do
 7:         src_container ← jobfiles.container_name
 8:         src_filename ← jobfiles.file_name
 9:         dest_container ← destWorkflowJobs[jobfiles.jobname]
10:         dest_filename← destWorkflowJobs[jobname].file_name
11:         src_cloud_file ← GETCLOUDFILE(src_container  src_filename )
12:         dest_cloud_file ← GETCLOUDFILE(dest_container  dest_filename )
13:         FileCounter ← FileCounter + 1
14:         if src_cloud_file.hash = dest_cloud_file.hash then
15:             ComparisonCounter ← ComparisonCounter + 1
16:     if FileCounter = ComparisonCounter then
17:         return True
18:     return False
```

Figure 4: Pseudocode to compare outputs produced by two given workflows.

5 RESULTS AND DISCUSSION

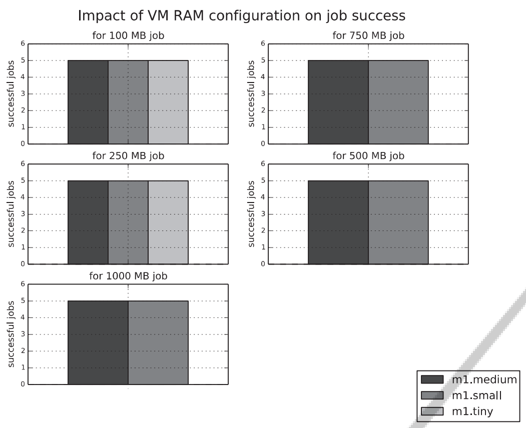

Figure 5: Cloud resource's RAM configuration impact on job success.

To demonstrate the effect of Cloud resource configuration such as RAM on job failure rate, a basic memory-consuming job is written in Java language. The job attempts to construct an alphabet string of given size (in MB), which is provided at runtime. To execute this experiment, three resource configurations, (a) m1.tiny, (b) m1.small and (c) m1.medium, each with 512 MB, 2048 MB and 4096 MB RAM respectively were used. Each job is executed at least 5 times with a given memory requirement on each resource configuration. The result in Figure 5 shows that jobs fail if required RAM (hardware) requirement is not fulfilled. All jobs with RAM requirement less than 500 MB executed successfully on all resource configurations. However, the jobs start to fail on Cloud resources with m1.tiny configuration as soon as the job's memory requirement approaches 500 MB because the jobs could not find enough available memory on the given resource. This result confirms the presented argument (discussed in section 1 and also in section 3) regarding the need for collecting Cloud resource configuration and its impact on job failure. Since a workflow is composed of many jobs, which are executed in a given order, a single job failure can result in a workflow execution failure. Therefore, collecting Cloud-aware provenance is essential for reproducing a scientific workflow execution on the Cloud.

To evaluate the presented mapping algorithm, which collects the Cloud infrastructure information and interlinks it with the workflow provenance, a Python based prototype has been developed using Apache Libcloud (Apache Libcloud – http://libcloud.apache.org), a library to interact with the Cloud middleware. The presented evaluation of the prototype is very basic currently. However, as this work progresses further a full evaluation will be conducted. To evaluate this prototype, a 20 core Cloud infrastructure is acquired from the Open Science Data Cloud (OSDC) (https://www.opensciencedatacloud.org/). This Cloud infrastructure uses the OpenStack middleware (openstack.org) to provide infrastructure-as-a-Service capability. A small Condor cluster of three virtual machines is also configured. In this cluster, one machine is a master node, which is used to submit workflows, and the remaining two are compute nodes. These compute nodes are used to execute workflow jobs. Using the Pegasus APIs, a basic *wordcount* workflow application composed of four jobs is written. This workflow has both control and data dependencies (Ramakrishnan and Plale, 2010) among its jobs, which is a common characteristic in scientific workflows. The first job (*Split* job) takes a text file and splits it into two files of almost equal length. Later, two jobs (*Analysis* jobs), each take one file as input, and then calculate the number of words in the given file. The fourth job (*merge* job) takes the outputs of earlier analysis jobs and calculates the final result i.e. total number of words in both files.

This workflow is submitted using Pegasus. The wfID assigned to this workflow is 114. The collected Cloud resource information is stored in database. Table I. shows the provenance mapping records in the Provenance Store for this workflow. The collected information includes the *flavour* and *image* (*image name* and *Image id*) configuration parameters. The *Image id* uniquely identifies an OS image hosted on the Cloud and this image contains all the software or libraries used during the job execution. As an image contains all the required libraries of a job, this prototype does not extract the installed libraries information from the virtual machine at the moment for workflow reproducibility purpose. However, this can be done in future iterations to enable the proposed approach to reconfigure a resource at runtime on the Cloud.

The reproducibility of the workflow using the proposed approach (discussed in section 4.2) has also been tested. The prototype is requested to repeat the workflow with wfID 114.

Upon receiving the request, it first collects the resource configurations, captured from earlier execution, from the database and provisions the resources on the Cloud infrastructure. The name of re-provisioned resource(s) for the repeated workflow has a postfix '-rep.novalocal' e.g. *mynova-*

Table 1: Cloud infrastructure mapped to the jobs of workflow with ID 114.

wfID	Host IP	nodename	Flavour Id	minRAM (MB)	minHD (GB)	vCPU	Image name	Image id
114	172.16.1.49	osdc-vm3.novalocal	2	2048	20	1	wf_peg_repeat	f102960c- 557c-4253-8277-2df5ffe3c169
114	172.16.1.98	mynode.novalocal	2	2048	20	1	wf_peg_repeat	102960c- 557c-4253-8277-2df5ffe3c169

Table 2: Cloud infrastructure information of repeated workflow (wfIDs: 117 and 122) after repeating workflow 114.

wfID	Host IP	nodename	Flavour Id	minRAM (MB)	minHD (GB)	vCPU	Image name	Image id
117	172.16.1.183	osdc-vm3-rep.novalocal	2	2048	20	1	wf_peg_repeat	f102960c- 557c-4253-8277-2df5ffe3c169
117	172.16.1.187	mynode-rep.novalocal	2	2048	20	1	wf_peg_repeat	f102960c- 557c-4253-8277-2df5ffe3c169
122	172.16.1.114	osdc-vm3-rep.novalocal	2	2048	20	1	wf_peg_repeat	f102960c- 557c-4253-8277-2df5ffe3c169
122	172.16.1.112	mynode-rep.novalocal	2	2048	20	1	wf_peg_repeat	f102960c- 557c-4253-8277-2df5ffe3c169

Table 3: Comparing outputs produced by workflows 114 (original workflow) and 117 (repeated workflow).

Job	WF ID	Container Name	File Name	MD5 Hash
Split	114	wfoutput123011	wordlist1	0d934584cbc124eed93c4464ab178a5d
	117	wfoutput125819	wordlist1	0d934584cbc124eed93c4464ab178a5d
	114	wfoutput123011	wordlist2	1bc6ffead85bd62b5a7a1be1dc672006
	117	wfoutput125819	wordlist2	1bc6ffead85bd62b5a7a1be1dc672006
Analysis 1	114	wfoutput123011	analysis1	494f24e426dba5cc1ce9a132d50ccbda
	117	wfoutput125819	analysis1	494f24e426dba5cc1ce9a132d50ccbda
Analysis 2	114	wfoutput123011	analysis2	127e8dbd6beffdd2e9dfed79d46e1ebc
	117	wfoutput125819	analysis2	127e8dbd6beffdd2e9dfed79d46e1ebc
Merge	114	wfoutput123011	merge_output	d0bd408843b90e36eb8126b397c6efed
	117	wfoutput125819	merge_output	d0bd408843b90e36eb8126b397c6efed

rep.novalocal as shown in Table 2. It was named *mynova.novalocal* in original workflow execution as shown in Table 1. From Table 2, one can assess that similar resources have been re-provisioned using the proposed approach to reproduce the workflow execution because the RAM, Hard disk, vCPUs and image configurations are similar to the resources used for workflow with wfID 114 (as shown in Table 1). This preliminary evaluation confirms that the similar resources on the Cloud can be re-provisioned with the Cloud-aware provenance information collected using the proposed approach (discussed previously in section 4). Table 2 shows two repeated workflow instances of original workflow 114.

The other aspect to evaluate the workflow reproducibility (as discussed in section 3) is to compare the outputs produced by both workflows. This has been achieved using the algorithm presented in Figure 4 (discussed in section 4.3). Four jobs in both the given workflows i.e. 114 and 117 produce the same number of output files (see Table 3). The *Split* job produces two output files i.e. *wordlist1* and *wordlist2*. Two analysis jobs, *Analysis1* and *Analysis2*, consume the wordlist1 and wordlist2 files, and produce the *analysis1* and *analysis2* files respectively. The merge job consumes the *analysis1* and *analysis2* files and produces the *merge_output* file. The hash values of these files are shown in the MD5 Hash column of the Table 3, here both given workflows are compared with each other. For instance, the hash value of *wordlist1* produced by the *Split* job of workflow 117 is compared with the hash value of *wordlist1* produced by the *Split* job of workflow 114. If both the hash values are same, the algorithm returns true. This process is repeated for all the files produced by both workflows. The algorithm confirms the verification of workflow outputs if the corresponding files in both workflows have the same hash values. Otherwise, the verification process fails.

6 CONCLUSION AND FUTURE DIRECTION

In this paper, the motivation and the issues related to workflow reproducibility due to workflow execution on the Cloud infrastructure have been identified. The dynamic nature of the Cloud makes provenance capturing of workflow(s) and their underlying execution environment(s) and their reproducibility a difficult challenge. A workflow reproducibility model (discussed in section 3) has been presented after analysing the literature and workflow execution scenario on the Cloud infrastructure. A proposed architecture has been presented that can augment the existing workflow provenance with the information of the Cloud infrastructure. Combining these two can assist in re-provisioning the similar execution environment to reproduce a workflow execution. The Cloud infrastructure information collection mechanism has been presented in this paper in section 4.1. This mechanism iterates over the workflow jobs and establishes mappings with the resource information available on the Cloud. This job to Cloud resource mapping can then be used to reproduce a workflow execution. The process of reproducing a workflow execution with the proposed approach has been discussed in section 4.2. In this paper, the workflow reproducibility is verified by comparing the outputs produced by the workflows. An algorithm has been discussed in section 4.3 (see Figure 4) that compares the outputs produced by two given workflows. A python-based prototype was developed for evaluating the proposed approach. The results show that the proposed approach can capture the Cloud-aware provenance information (as discussed in section 4) by collecting the information related to Cloud infrastructure (virtual machines) used during a workflow execution. It can then provision a similar execution infrastructure i.e. same resource configuration on the Cloud using the Cloud-aware provenance information to reproduce a workflow execution. In future, the proposed approach will be extended and a detailed evaluation of the proposed approach will be conducted. Different performance matrices such as the impact of the proposed approach on workflow execution time, impact of different resource configuration on workflow execution performance, and total resource provision time will also be measured. In this paper, only workflow outputs have been used to compare two workflows' provenance traces. In future, the comparison algorithm will also incorporate workflow structure and execution infrastructure (as discussed in section 3) to verify workflow reproducibility. The proposed approach has not addressed the issue of securing the stored Cloud-aware provenance. In future, the presented architecture will be extended by adding a security layer on top of the collected Cloud-aware provenance.

ACKNOWLEDGEMENTS

This research work has been funded by a European Union FP-7 project, N4U – neuGrid4Users. This project aims to assist the neuro-scientific community in analysing brain scans using workflows and distributed infrastructure (Grid and Cloud) to identify biomarkers that can help in diagnosing the Alzheimer disease. Besides this, the support provided by OSDC by offering a free Cloud infrastructure of 20 cores is highly appreciated. Such public offerings can really benefit research and researchers who are short of such resources.

REFERENCES

(2014). GriPhyN: http://www.phys.utb.edu/griphyn/ [Last visited 30-12-2014].

(2014). SDSS: http://www.sdss.org [Last visited 30-12-2014].

Azarnoosh, S., Rynge, M., Juve, G., Deelman, E., Niec, M., Malawski, M., and da Silva, R. (2013). Introducing precip: An api for managing repeatable experiments in the cloud. In 5th IEEE Conference on Cloud Computing Technology and Science (CloudCom), volume 2, pages 19–26.

Belhajjame, K., Roos, M., Garcia-Cuesta, E., Klyne, G., Zhao, J., De Roure, D., Goble, C., Gomez-Perez, J. M., Hettne, K., and Garrido, A. (2012). Why workflows break - understanding and combating decay in taverna workflows. In Proceedings of the 2012 IEEE 8th International Conference on E-Science (e-

Science'12), pages 1–9, USA. IEEE Computer Society.

Stodden, V. C. (2010). Reproducible research: Addressing the need for data and code sharing in computational science. Computing in Science & Engineering, 12.

Chirigati, F., Shasha, D., and Freire, J. (2013). Reprozip: Using provenance to support computational reproducibility. In Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance, TaPP '13, pages 1–4, Berkeley, USA. USENIX Association.

Oliveira, D., Ogasawara, E., Bai˜ao, F., and Mattoso, M. (2010). Scicumulus: A lightweight cloud middleware to explore many task computing paradigm in scientific workflows. In Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on, pages 378–385.

Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Patil, S., Su, M.-H., Vahi, K., and Livny, M. (2004). Pegasus: Mapping scientific workflows onto the grid. In Dikaiakos, M., editor, Grid Computing, volume 3165 of Lecture Notes in Computer Science, pages 11–20. Springer Berlin Heidelberg.

Deelman, E., Gannon, D., Shields, M., and Taylor, I. (2008). Workflows and e-science: An overview of workflow system features and capabilities.

Foster, I. and Kesselman, C., editors (1999). The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers Inc., USA.

Foster, I., V¨ockler, J., Wilde, M., and Zhao, Y. (2002). Chimera: a virtual data system for representing, querying, and automating data derivation. In Scientific and Statistical Database Management, Proceedings. 14th International Conference on, pages 37–46.

Foster, I., Zhao, Y., Raicu, I., and Lu, S. (2008). Cloud computing and grid com- puting 360-degree compared. In Grid Computing Environments Workshop, 2008. GCE '08, pages 1–10.

Groth, P., Deelman, E., Juve, G., Mehta, G., and Berriman, B. (2009). Pipeline- centric provenance model. In Proceedings of the 4th Workshop on Work-flows in Support of Large-Scale Science, WORKS '09, pages 4:1–4:8, USA. ACM.

Howe, B. (2012). Virtual appliances, cloud computing, and reproducible re- search. Computing in Science Engineering, 14(4):36–41.

Janin, Y., Vincent, C., and Duraffort, R. (2014). Care, the comprehensive archiver for reproducible execution. In Proceedings of the 1st ACM SIG- PLAN Workshop on Reproducible Research Methodologies and New Publication Models in Computer Engineering, TRUST '14, pages 1:1–1:7, USA. ACM.

Juve, G. and Deelman, E. (2010). Scientific workflows and clouds. Crossroads, 16(3):14–18.

Kim, J., Deelman, E., Gil, Y., Mehta, G., and Ratnakar, V. (2008). Provenance trails in the wings-pegasus system. Concurr. Comput. : Pract. Exper., 20(5):587–597.

Ko, R., Lee, B., and Pearson, S. (2011). Towards achieving accountability, auditability and trust in cloud computing. In Advances in Computing and Communications, volume 193 of Communications in Computer and Information Science, pages 432–444. Springer Berlin Heidelberg.

Lifschitz, S., Gomes, L., and Rehen, S. K. (2011). Dealing with reusability and reproducibility for scientific workflows. In Bioinformatics and Biomedicine Workshops (BIBMW), 2011 IEEE International Conference on, pages 625–632. IEEE. 38, 69.

Macko, P., Chiarini, M., and Seltzer, M. (2011). Collecting provenance via the xen hypervisor. 3rd USENIX Workshop on the Theory and Practice of Provenance (TAPP).

Mehmood, Y., Habib, I., Bloodsworth, P., Anjum, A., Lansdale, T., and McClatchey, R. (2009). A middleware agnostic infrastructure for neuro- imaging analysis. In Computer-Based Medical Systems, 2009. CBMS 2009. 22nd IEEE International Symposium on, pages 1–4.

Mei, L., Chan, W. K., and Tse, T. H. (2008). A tale of clouds: Paradigm comparisons and some thoughts on research issues. In Proceedings of the 2008 IEEE Asia-Pacific Services Computing Conference, APSCC '08, pages 464–469, USA. IEEE Computer Society.

Mell, P. M. and Grance, T. (2011). Sp 800-145. the nist definition of cloud computing. Technical report, Gaithersburg, MD, United States.

Missier, P., Woodman, S., Hiden, H., and Watson, P. (2013). Provenance and data differencing for workflow reproducibility analysis. Concurrency and Computation: Practice and Experience.

Munir, K., Kiani, S. L., Hasham, K., McClatchey, R., Branson, A., and Sham- dasani, J. (2013). An integrated e-science analysis base for computation neuroscience experiments and analysis. Procedia - Social and Behavioral Sciences, 73(0):85 – 92. Proceedings of the 2nd International Conference on Integrated Information (IC-ININFO 2012), Budapest, Hungary, August 30 – September 3, 2012.

Munir, K., Liaquat Kiani, S., Hasham, K., McClatchey, R., Branson, A., and Shamdasani, J. (2014). Provision of an integrated data analysis platform for computational neuroscience experiments. Journal of Systems and In-formation Technology, 16(3):150–169.

Ramakrishnan, L. and Plale, B. (2010). A multi-dimensional classification model for scientific workflow characteristics. In Proceedings of the 1st International Workshop on Workflow Approaches to New Data-centric Science, Wands '10, pages 4:1–4:12, USA. ACM.

Roure, D. D., Manuel, J., Hettne, K., Belhajjame, K., Palma, R., Klyne, G., Missier, P., Ruiz, J. E., and Goble, C. (2011). Towards the preservation of scientific workflows. In Procs. of the 8th International Conference on Preservation of Digital Objects (iPRES 2011). ACM.

Sandve, G. K., Nekrutenko, A., Taylor, J., and Hovig, E. (2013). Ten sim- ple rules for reproducible computational research. PLoS Comput Biol, 9(10):e1003285.

Santana-Perez, I., Ferreira da Silva, R., Rynge, M., Deelman, E., P´erez- Hern´andez, M., and Corcho, O.

(2014a). A semantic-based approach to attain reproducibility of computational environments in scientific work- flows: A case study. In Parallel Processing Workshops, volume 8805 of Lecture Notes in Computer Science, pages 452–463. Springer International Publishing.

Santana-Perez, I., Ferreira da Silva, R., Rynge, M., Deelman, E., Perez- Hernandez, M. S., and Corcho, O. (2014b). Leveraging semantics to improve reproducibility in scientific workflows. In The reproducibility at XSEDE workshop.

Scheidegger, C., Koop, D., Santos, E., Vo, H., Callahan, S., Freire, J., and Silva, C. (2008). Tackling the provenance challenge one layer at a time. Concurr. Comput. : Pract. Exper., 20(5):473–483.

Shamdasani, J., Branson, A., and McClatchey, R. (2012). Towards semantic provenance in cristal. In Third International Workshop on the role of Se- mantic Web in Provenance Management (SWPM 2012).

Simmhan, Y. L., Plale, B., and Gannon, D. (2005). A survey of data provenance in e-science. SIGMOD Rec., 34(3):31–36.

SMS, C., CE, P., D, O., MLM, C., and M., M. (2011). Capturing distributed provenance metadata from cloud-based scientific workflows. Information and Data Management, 2:43–50.

Stallings, W. (2010). Cryptography and Network Security: Principles and Prac- tice. Prentice Hall Press, Upper Saddle River, NJ, USA, 5th edition.

Stevens, R. D., Robinson, A. J., and Goble, C. A. (2003). myGrid: personalised bioinformatics on the information grid, Bioinformatics, 19:i302–i304.

Tan, Y. S., Ko, R. K., Jagadpramana, P., Suen, C. H., Kirchberg, M., Lim, T. H., Lee, B. S., Singla, A., Mermoud, K., Keller, D., and Duc, H. (2012). Tracking of data leaving the cloud. 2013 12th IEEE International Confer- ence on Trust, Security and Privacy in Computing and Communications, 0:137–144.

Tannenbaum, T., Wright, D., Miller, K., and Livny, M. (2002). Beowulf cluster computing with linux. chapter Condor: A Distributed Job Scheduler, pages 307–350. MIT Press, Cambridge, MA, USA.

Vouk, M. (2008). Cloud computing #x2014; issues, research and implementa- tions. In Information Technology Interfaces, 2008. ITI 2008. 30th Interna- tional Conference on, pages 31–40.

Zhang, O. Q., Kirchberg, M., Ko, R. K., and Lee, B. S. (2011). How to track your data: The case for cloud computing provenance. In Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on, pages 446–453. IEEE.

Zhao, X., Zhang, Y., Wu, Y., Chen, K., Jiang, J., and Li, K. (2014). Liquid: A scalable deduplication file system for virtual machine images. Parallel and Distributed Systems, IEEE Transactions on, 25(5):1257–1266.

Zhao, Y., Fei, X., Raicu, I., and Lu, S. (2011). Opportunities and challenges in running scientific workflows on the cloud. In Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2011 International Con- ference on, pages 455–462.