Re-Provisioning of Cloud-based Execution Infrastructure using the Cloud-Aware Provenance to Facilitate Scientific Workflow Execution Reproducibility

Khawar Hasham*, Kamran Munir, Richard McClatchey, and Jetendr Shamdasani

Centre for Complex Cooperative Systems (CCCS), Department of Computer Science and Creative Technologies (CSCT), University of the West of England (UWE), Frenchay Campus, Coldharbour Lane, Bristol, BS16 1QY, United Kingdom
{mian.ahmad,kamran2.munir,richard.mcclatchey,jetendr2.shamdasani}@uwe.ac.uk

Abstract. Provenance has been considered as a means to achieve scientific workflow reproducibility to verify the workflow processes and results. Cloud computing provides a new computing paradigm for the workflow execution by offering a dynamic and scalable environment with on-demand resource provisioning. In the absence of Cloud infrastructure information, achieving workflow reproducibility on the Cloud becomes a challenge. This paper presents a framework, named ReCAP, to capture the Cloud infrastructure information and to interlink it with the workflow provenance to establish the Cloud-Aware Provenance (CAP). This paper identifies different scenarios of using the Cloud for workflow execution and presents different mapping approaches. The reproducibility of the workflow execution is performed by re-provisioning the similar Cloud resources using CAP and re-executing the workflow; and by comparing the outputs of workflows. Finally, this paper also presents the evaluation of ReCAP in terms of captured provenance, workflow execution time and workflow output comparison.

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

1 Introduction

Modern scientific experiments such as the Large Hadron Collider (LHC)[1], and projects such as neuGRID [1] and its follow-on neuGRIDforUsers [2] are producing huge amounts of data. This data is processed and analysed to extract meaningful information by employing scientific workflows that orchestrate the complex data analysis processes [3]. A large pool of compute and data resources

* This is the corresponding author

[1] http://lhc.web.cern.ch

Citation:
Hasham, K; Munir, K; McClatchey, R; Shamdasani, J, 2016, "Re-provisioning of Cloud-Based Execution Infrastructure Using the Cloud-Aware Provenance to Facilitate Scientific Workflow Execution Reproducibility", Edited by: Helfert, M; Munoz, VM; Ferguson, D, Cloud Computing and Services Science, CLOSER 2015, Springer Book Series: Communications in Computer and Information Science, Volume: 581, Pages: 74-94, DOI: 10.1007/978-3-319-29582-4_5, Published: 03 February 2016

is required to execute the workflows. These resources have been available through the Grid [4] and are now also being offered by the Cloud-based infrastructures.

Cloud computing [5] offers a new computing and storage paradigm, which is dynamically scalable and usually works on a pay-as-you-go cost model. Its ability to provide an on-demand computing infrastructure enables distributed processing of scientific workflows with increased complexity and data requirements [6]. Research is under way to exploit the potential of Cloud infrastructure for workflow execution [7].

During the data processing, an important consideration is given to collect provenance [8] information. This can provide detailed information about both the inputs and the processed outputs, and the processes involved in a workflow execution. This information can be used to debug the workflow execution, 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 [9]. 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 [10] 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 lack of 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 on a resource with more available Random Access Memory (RAM) because it can accommodate more data in RAM, which is a faster medium to access data than hard disk. However, the job's performance will degrade if the allotted resource does not provide adequate RAM. Moreover, it is also possible that jobs will 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 nature of Cloud computing makes the capturing and processing of provenance information a major research challenge [11, 12]. Since Cloud presents a transparent access to dynamic execution resources, the workflow parameters including execution resource configuration should also be known to a scientist [13] 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 (e.g. [14]) and a few initiatives (e.g. [15, 16]) 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 framework, named ReCAP, that augments workflow provenance with the Cloud infrastructure information; and uses it to provision similar execution environment(s) and reproduces the execution of a given workflow. Important areas discussed in this paper are as follows: section 2 presents the related work in provenance related systems. Section 3 presents a set of requirements identified for workflow reproducibility on the Cloud after collecting guidelines used and discussed in literature. Section 4 presents an overview of ReCAP's architecture. Section 4 also discusses two scenarios of using Cloud resources and the provenance capturing approaches devised for these scenarios. 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 [17, 18] has been carried out in workflow provenance for Grid-based workflow management systems. Chimera [17] is designed to manage the data-intensive analysis for high-energy physics (GriPhyN)[2] and astronomy (SDSS)(http://www.sdss.org) 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 [18] 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 [19] 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 (e.g. [15, 16]) 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 [20, 21] or hypervisor level [22]. 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 [9] 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 [23, 24] and research studies e.g. [25] on collecting provenance and using it to reproduce an experiment. A semantic-based approach [25] has been proposed to improve reproducibility of workflows in the Cloud. This approach uses ontologies to extract information about the

[2] http://www.phys.utb.edu/griphyn/

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 [23] 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. Similarly, a Linux- based tool, CARE [24], 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. Both these approach are useful at individual job level but are not applicable to an entire workflow, which is the focus of this paper. Moreover, they do not maintain the hardware configuration of the underlined execution machine. Furthermore, these approaches operate along with the job on the virtual machine (VM). On the contrary, out proposed approach works outside the virtual machine and therefore does not interfere with job execution.

3 Requirements for Workflow Reproducibility on Cloud

As per our understanding of the literature, there is not a standard reproducibility model proposed thus far for scientific workflows, especially in a Cloud environment. However, there are some guidelines or policies, which have been highlighted in literature to reproduce experiments. There is one good effort [26] in this regard, but it mainly talks about reproducible papers and it does not consider execution environment of workflows. In this section, we have highlighted a set of requirements for workflow reproducibility on Cloud that can provide guidelines for future work in this regard. These requirements are discussed as follows.

- **Data and Code Sharing:** In computational science, particularly for scientific workflow executions, it is emphasized that the data, code, and the workflow description should be available in order to reproduce an experiment [27]. Code must be available to be distributed, and data must be accessible in a readable format [28]. In the absence of such information, experiment reproducibility cannot be achieved because different result would be produced if the input data changes. It is also possible that the experiment cannot be successfully executed in the absence of the required code and its dependencies or configurations.
- **Execution Infrastructure:** The execution infrastructure is composed of a set of computational resources (e.g. execution nodes, storage devices, networking). The physical approach, where actual computational hardware are made available for long time periods to scientists, often conserves the computational environment including supercomputers, clusters, or Grids [25]. As a result, scientists are able to reproduce their experiments in the same hardware environment. However, this luxury is not available in the Cloud

in which resources are virtual and dynamic. Therefore, it is important to collect the Cloud resource information in such a manner that will assist in re-provisioning of similar resources on the Cloud for workflow re-execution.

From a resource provisioning as well as a performance point of view, various factors such as RAM, vCPU, Hard Disk and CPU speed (e.g. MIPS) are important in selecting appropriate resources especially on the Cloud. As discussed previously, the RAM can affect the job's execution performance as well as its failure rate. A job will fail if it is scheduled to a resource with less available RAM (as shown in Figure 4). Similarly, vCPU (virtual CPUs meaning CPU cores) along with the MIPS (million instructions per second) value directly affect the job execution performance. In a study [29], it was found that the workflow task durations differ for each major Cloud, despite the identical setup.

Hard disk capacity also becomes an important factor in provisioning a new resource on the Cloud. It was argued [29] that building images for scientific applications requires adequate storage within a virtual machine (VM). In addition to the OS and the application software, this storage is used to hold job inputs and output that are consumed and produced by a workflow job executing on a VM [29].

- **Software Environment:** Apart from knowing the hardware infrastructure, it is also essential to collect 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 [30] 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 [31]. 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 also retrieves the image information to build a virtual machine on which the workflow job was executed.
- **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 [32]. 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 following levels:

1. 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.
2. Execution infrastructure (software environment, resource configuration) used on the Cloud for a workflow execution should also be compared.
3. 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 [33], which perform workflow provenance comparison to determine differences in reproduced workflows. The proposed system in this paper incorporates the workflow output comparison to determine the reproducibility of a workflow.

– **Cloud Resource Pricing:** Cloud resource pricing can be important for experiments in which cost is also a main factor. However, this can also be argued that this information is not trivial for an experiment due to strong industry competition between big Cloud providers such as Amazon, Google, Microsoft etc., which can bring prices down. Having said this, one still cannot deny the fact that a cost is associated with each acquired resource on the Cloud, thus making this factor important to be focused on. The pricing factor has been used in various studies to conduct the feasibility of a Cloud environment for workflow execution [6]. In this study, the cost factor for various resources such as compute and storage has been evaluated for workflow execution. The pricing information has also been used in cost-effective scheduling and provisioning algorithms [34, 35]. Therefore, this pricing information, if collected as part of provenance, can help in reproducing an experiment within the similar cost as was incurred in earlier execution. However, one must keep this in mind that the prices are dynamic and subject to change and it depends entirely on the Cloud providers. For an environment, in which cost does not change rapidly, such information can be helpful. Therefore, this information is captured as part of the Cloud-Aware Provenance data.

Workflow versioning is another factor that aids in achieving workflow reproducibility [36]. Sandve et al. [26] also suggested archiving the exact versions of all processes and enabling version control on all scripts used in an experiment. With the help of workflow versioning, a user can track the evolution of a workflow itself. Since the focus of this research work is on the workflow execution provenance and not on the workflow evolution, this factor is outside the scope of the presented work. Based on the identified factors in this section, following section presents a framework, named ReCAP, to capture the Cloud infrastructure information and to interlink it with the workflow provenance to establish the Cloud-Aware Provenance (CAP). This information is used to re-provision similar execution infrastructure on the Cloud in order to reproduce the execution of a scientific workflow.

4 ReCAP: Workflow Reproducibility using Cloud-Aware Provenance

An overview of the ReCAP's architecture, a proposed solution, is presented in this section. This architecture is inspired by the mechanism used in a paper [37] for executing workflows on the Cloud. Figure 1 illustrates the proposed architecture that collects the Cloud infrastructure information and interlinks it with the workflow provenance gathered 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 (CAP). The components of this architecture are discussed as follow:

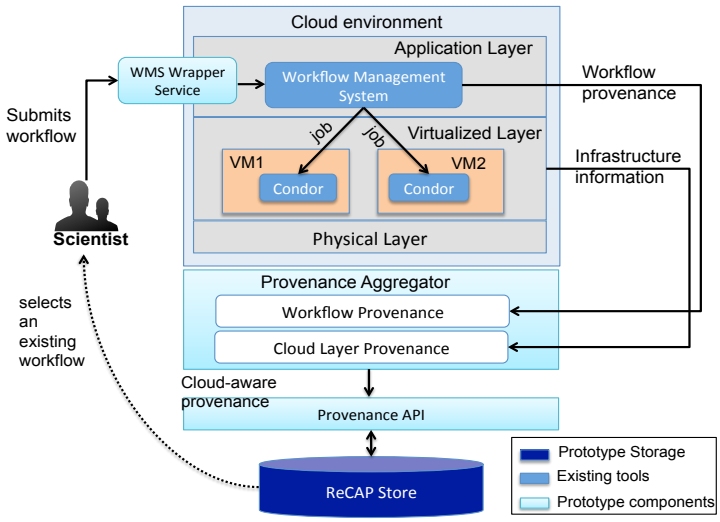

Fig. 1. General overview of the ReCAP architecture

– **WMS Wrapper Service:** This component exposes the functionality of an underlining workflow management system (WMS) by providing a wrapper service. It is responsible for receiving various user and ReCAP's components requests in submitting a user provided workflow and monitoring its status. For instance, there is no suitable HTTP-based facility available that a user can use to submit a workflow and its associated files to Pegasus. Traditionally, a command-based approach is used in which Pegasus provided commands are invoked from a terminal. With such a service based component, a user can submit his workflow through an HTTP client. Another purpose of this component is to engage with a user from the very first step of workflow execution i.e. workflow submission. Although this paper is focusing on workflow execution, it still needs a mechanism to access the submitted workflow and its associated configuration files in order for it to reproduce and

resubmit the same workflow. Therefore, such a mechanism was required that can act as an entry point for the system and also help in ensuring the access to the workflow source, which is one of the points of the reproducibility requirements identified for the Cloud (see Section 3).

- **Workflow Provenance:** This component, named *WFProvenance*, is responsible for receiving provenance captured at the application level by the workflow management system e.g. 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, *CloudLayerProvenance*, 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 interacts with the Cloud infrastructure as an outside client to obtain the resource configuration information. This component is discussed in detail in Section 4.2 and 4.3.
- **Provenance Aggregator:** This is the main component task 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 *ProvenanceAPI*.
- **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 *ProvenanceAggregator* would interact with it to store the workflow provenance information. This approach gives flexibility to implement authentication or authorization in accessing the provenance store.
- **ReCAP Store:** This data store is designed to keep record of the workflow and its related configuration files being used to submit a user analysis on the Cloud. It also keeps the mapping between workflow jobs and the virtual resources used for execution on the Cloud infrastructure. This information is later retrieved to reproduce the workflow execution.

4.1 Cloud Usage Scenarios

This section discusses the job to Cloud resource mapping, which will be used later for re-executing a workflow on similar Cloud resources, mechanisms devised in this research study. Before indulging into detailed discussion of these mechanisms, first it is important to understand two different resource usage scenarios on Cloud. These scenarios and their understanding provide a better picture of the requirements and the motivation behind devising different mechanisms to establish job to Cloud resource mapping for each discussed scenario.

– **Static Environment on Cloud**
  In this environment, the virtual resources, once provisioned, remain in RUNNING state on Cloud for a longer time. This means that the resources will be accessible even after a workflows execution is finished. This environment is similar to creating a virtual cluster or Grid on top of Clouds resources. Such a Cloud environment is also in used in the N4U infrastructure. The Static Mapping approach devised for such environment has been discussed in Section 4.2.
– **Dynamic Environment on Cloud**
  In this environment, resources are provisioned on demand and released when they are no more required. This means that the virtual machines are shutdown after the job is done. Therefore, a virtual resource, which was used to execute a job, will not be accessible once a job is finished. The Eager Mapping has been devised (see Section 4.3) to handle this scenario.

The mapping approaches discussed in following sections achieve the job to Cloud resource mapping using the workflow provenance information. One such information is an indication of execution host or its IP in the collected workflow provenance. Many a workflow management systems such as Pegasus, VisTrail or Chiron [38] do maintain either machine name or IP information. In Clouds infrastructure layer across one Cloud provider or for one user, no two virtual machines can have same IP at any given time. This means any running virtual machine should have unique IP or name. However, it is possible that a name or IP can be reused later for new virtual machines. All rest properties of a virtual machine accessible through the infrastructure layer can be used by multiple machines at a time. For instance, multiple machines can be provisioned with flavour m1.small or with OS image Ububtu 14.04 or Fedora etc.

4.2 Static Mapping Approach

As mentioned earlier, this information is used for reprovisioning the resources to provide a similar execution infrastructure to repeat a workflow execution. The Static Mapping approach has been devised for the Static environment on the Cloud. 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 *ReCAPStore*. The flowchart of this mechanism is presented in Figure 2. 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

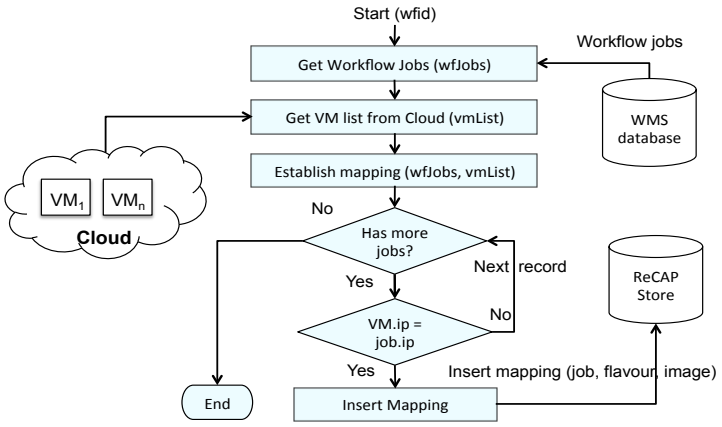

Fig. 2. Flowchart of the job to the Cloud resource mapping in the Static environment

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.

4.3 Eager Mapping Approach

This approach is devised to establish a job to Cloud resource mapping for the dynamic environment on Cloud. As discussed in Section 4.1, the resource on Cloud may not be accessible once a job is finished, thus making a job to resource mapping a challenge. This is why, this approach attempts to identify, as early as possible, the virtual machine on which a job is executing. In this mapping approach, the Cloud-aware provenance is acquired in two phases, which are discussed as follows.

Phase 1: Temporary Job to Resource Mapping: In this phase, the Eager approach monitors the underlying WMS database i.e. Pegasus for the implemented prototype. In Pegasus, along with the host name, its database also maintains the Condor's *schedd_ID*, which is assigned to each job by Condor [39]. The monitoring thread in $WFProvenance$ retrieves the job's Condor ID and contacts the WMS Wrapper Service (WMS-WS) for information about the job. Since the WMS-WS works on top of the underlying workflow managment

system, it has an access to the Condor cluster. Upon receiving the request, WMS-WS retrieves job information from the Condor. This information contains the machine IP on which the job is currently running. The *CloudLayerProvenance* component retrieves the virtual machine's configuration information from the Cloud middleware based on the machine IP (as discussed in the Section 4.2) and stores this information in the database. This information is treated as temporary because the job is not finished yet and there is a possibility that a job may be re-scheduled to another machine due to runtime failures [40]. This information is then used in the second phase for establishing the final mapping between the job and the Cloud resource. The flowchart of this mechanism is presented in Figure 3.

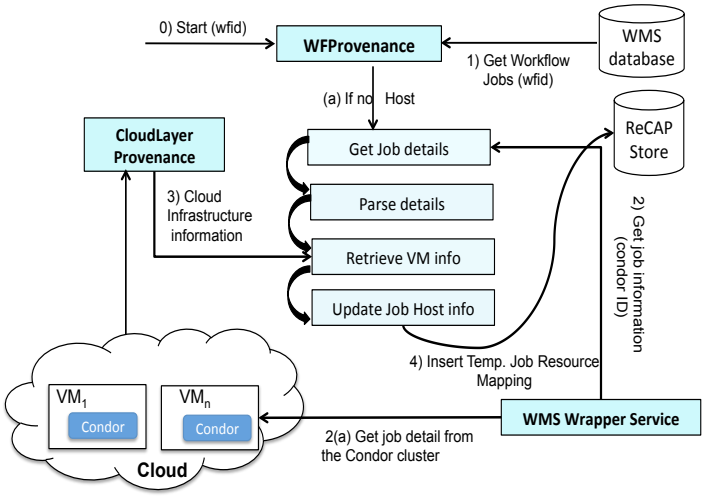

Fig. 3. Temporary resource mapping established in the phase 1 of the Eager approach

Phase 2: Final Job to Resource Mapping: This phase starts when the workflow execution is finished. The *ProvenanceAggregator* component starts the job to resource mapping process. In doing so, it retrieves the list of workflows from the database and list of virtual machines from the Cloud middleware through the *CloudLayerProvenance* component. It starts the mapping between the jobs and the virtual machines based on the IP information, stored in the database, associated with the jobs. In the case of not finding any host information in the database, which is possible in the Dynamic use case, the *ProvenanceAggregator* retrieves the resource information for that job from the temporary repository that was created in the first phase (as discussed in the Section 4.2). Upon finding the Cloud resource information, the *Provenance Aggregator* component registers this Cloud-Aware Provenance information in the *ReCAPStore*. Once the mapping for a job is established and stored in the database, its corresponding temporary mapping is removed in order to reduce

the disk storage overhead. The algorithm of the Eager mapping approach is shown in Algorithm 1.

Algorithm 1 Eager Approach

Require: $wfJobs$: Set of jobs in the workflow.

```
1: Phase 1
2: procedure JobMonitor(wfJobs)
3:     cloudResources ← getCloudResources(())}
4:     for all job ∈ wfJobs do
5:         condorid ← job.condor                          ▷ each job is assigned unique id
6:         hostname ←' pegasusClient.getHostInfo(condorid)
7:         vm ← cloudResource[hostname]
8:         if vm != None & vmMappingExists(vm, job) then
9:             resourceFlavor ← vm.flavor
10:            resourceImage ← vm.image
11:            createTempMapping(job, resourceFlavor, resourceImage)
12: Phase 2
13: procedure EstablishMapping(wfJobs)
14:     for all job ∈ wfJobs do
15:         vm ← getTempJobMapping(job)
16:         if tempMap then
17:             resourceImage ← vm.image
18:             resourceFlavor ← vm.image
19:             storeJobResourceMapping(job, resourceFlavor, resourceImage)
20:             removeTempMapping(job, resourceFlavor, resourceImage)
```

4.4 Workflow Reproducibility using ReCAP

In Section 4, different mapping approaches have been discussed to interlink the job to Cloud resource information, which 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 ReCAP to re-execute the workflow using the Cloud-aware provenance. ReCAP retrieves the given workflow from the *ReCAPStore* along with the Cloud resource mapping stored against this workflow. Using this mapping information, it retrieves the resource flavour and image configurations, and provisions the resources on Cloud. Once resources are provisioned, it submits the workflow for execution. 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 the execution of the workflow and start collecting its Cloud-aware provenance information. Recapturing the provenance of the repeated workflow is important, as this will enable us to verify the provisioned resources by comparing their resource configurations with the old resource configuration.

4.5 Workflow Output Comparison

Another aspect of workflow reproducibility 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 [41] 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.

Algorithm 2 Pseudocode to compare outputs produced by two given workflows

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
```

The proposed algorithm (as shown in Algorithm 2) 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 (http://swift.openstack.org), Amazon S3 (http://aws.amazon.com/s3) 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.

5 Results and Discussion

To demonstrate the effect of Cloud resource configuration such as RAM on job failure rate, a basic memory-consuming job is written in Java. 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 shown in Figure 4)as soon as the jobs 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.

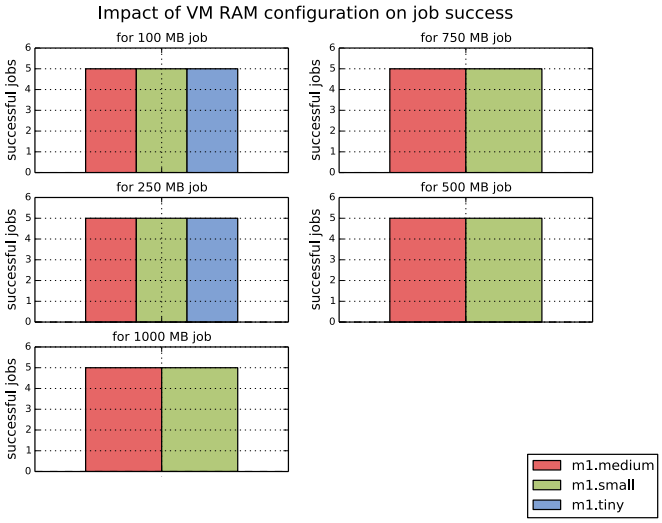

Fig. 4. The effect of the Cloud resource's RAM configuration on the job's success rate

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[3], 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 cores Cloud infrastructure is acquired from the Open Science Data Cloud (OSDC)(opensciencedatacloud.org). This Cloud infrastructure uses the OpenStack middleware (openstack.org) to

[3] http://libcloud.apache.org

provide Infrastructure-as-a-Service (IaaS) 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 wordcount workflow application composed of four jobs is written. This workflow has both control and data dependencies [42] among its jobs along with the split and merge characteristics, which are common characteristics 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 1 shows the provenance mapping records in the *ReCAPStore* 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 discussed earlier in Section 3). 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

Table 1. Cloud-Aware Provenance captured for a given workflow

WfID	Host IP	nodename	Flavour Id	minRAM	minHD	vCPU	image name	image Id
114	174.16.1.49	osdc-vm3	2	2048	20 GB	1	wf_peg_repeat	f102960c-557c-4253-8277-2df5ffe3c169
114	174.16.1.98	mynode	2	2048	20 GB	1	wf_peg_repeat	f102960c-557c-4253-8277-2df5ffe3c169

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 e.g. mynova-rep as shown in Table 2. It was named *'mynova'* in original workflow execution as shown in Table 1. From Table 2, one can assess that similar resources have been re-provisioned using the ReCAP system 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 result confirms that the similar resources on the Cloud can be re- provisioned with the Cloud-Aware Provenance (CAP) collected using the proposed approach (discussed in Section 4). Table 2 shows two repeated workflow instances of original workflow 114. In order to measure the execution time of the original workflow and the re- produced workflow on the similar execution infrastructure on the Cloud, the same

Table 2. Provenance data of the reproduced workflow showing that ReCAP successfully re-provisioned similar resources on the Cloud

WfID	Host IP	nodename	Flavour Id	minRAM	minHD	vCPU	image name	image Id
117	172.16.1.183	osdc-vm3-rep	2	2048	20 GB	1	wf_peg_repeat	f102960c-557c-4253-8277-2df5ffe3c169
117	172.16.1.187	mynode-rep	2	2048	20 GB	1	wf_peg_repea	f102960c-557c-4253-8277-2df5ffe3c169
122	172.16.1.114	osdc-vm3-rep	2	2048	20 GB	1	wf_peg_repeat	f102960c-557c-4253-8277-2df5ffe3c169
122	172.16.1.112	mynode-rep	2	2048	20 GB	1	wf_peg_repea	f102960c-557c-4253-8277-2df5ffe3c169

workflow was executed multiple times on the Cloud infrastructure. An average execution time is calculated for these workflow executions and treated as the average execution time of the original workflow. The ReCAP approach is then used to reproduce the workflow execution by re-provisioning the similar execution infrastructure using the Cloud-Aware Provenance (CAP). The same workflow was re-executed on the re-provisioned resources to measure the execution time of the reproduced workflow. Figure 5 shows the average workflow execution times for both the original and reproduced workflows respectively. In the case of original execution, the average workflow execution is 434.84 ± 6.52 seconds and the workflow execution time for the reproduced workflow is 434.76 ± 7.3657 seconds. This result shows that there is no significance difference (i.e. 0.08 seconds) in workflow execution time because of the similar execution infrastructure used for workflow re-execution. This result confirms that workflow can be reproduced with similar execution performance provided a similar execution infrastructure is available on the Cloud.

Fig. 5. Comparing the average workflow execution time of the original and the reproduced workflow execution

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 discussed in Section 4.5. 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. Table 3 shows that both workflows have produced the identical files because the hash values are same. In order to measure the impact of prove-

Table 3. Provenance data of the reproduced workflow showing that ReCAP successfully re-provisioned similar resources on the Cloud

Job	WfID	Container Name	File Name	MD5 Hash
Split	114	wfoutput123011	wordlist1	0d934584cbc124eed93c4464ab178a5d
	117	wfoutput125819	wordlist1	0d934584cbc124eed93c4464ab178a5d
	114	wfoutput123011	wordlist2	0d934584cbc124eed93c4464ab178a5d
	117	wfoutput125819	wordlist2	0d934584cbc124eed93c4464ab178a5d
Analysis1	114	wfoutput123011	analysis1	494f24e426dba5cc1ce9a132d50ccbda
	117	wfoutput125819	analysis1	494f24e426dba5cc1ce9a132d50ccbda
Analysis2	114	wfoutput123011	analysis2	127e8dbd6beffdd2e9dfed79d46e1ebc
	117	wfoutput125819	analysis2	127e8dbd6beffdd2e9dfed79d46e1ebc
Merge	114	wfoutput123011	merge_output	d0bd408843b90e36eb8126b397c6efed
	117	wfoutput125819	merge_output	d0bd408843b90e36eb8126b397c6efed

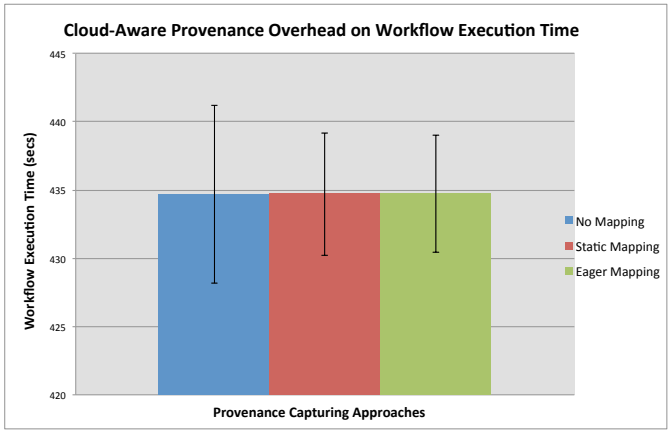

Fig. 6. Cloud-Aware Provenance capturing overhead on the workflow execution time

nance mapping approaches (as discussed in Section 4.2 and 4.3) on the workflow execution performance, the workflow jobs were modified to eliminate the effect of the data transfer time on the workflow execution. The jobs in the workflow mimic the job processing by introducing a sleep interval for a given time pe-

riod, which is passed as an argument. Figure 6 shows that the average workflow execution time in the absence of any provenance approach (i.e. No Mapping) is 434.69 ± 6.52 seconds. The average workflow execution time is 434.71 ± 4.49 and 434.74 ± 4.28 seconds for the Static and Eager Mapping approaches respectively. The difference between the execution times is 0.02 and 0.05 seconds for the Static and Eager approaches respectively. This slight difference in execution time is mainly caused by the delays a job faces during its execution. The overall workflow execution time remains almost the same in the presence of the proposed provenance capturing approaches. The main reason for these mapping approaches to not having a major impact on the workflow execution time is because they work outside the virtual machines, thus they don't interfere with the job execution.

6 Conclusion and Future Direction

The dynamic nature of the Cloud makes provenance capturing of workflow(s) with the underlying execution environment(s) and their reproducibility a difficult challenge. In this regard, a list of workflow reproducibility requirements has been presented after analysing the literature and workflow execution scenario on the Cloud infrastructure. The proposed ReCAP's framework can augment the existing workflow provenance with the Cloud infrastructure information. Based on the identified Cloud usage scenarios i.e. Static and Dynamic , the proposed mapping approaches iterate over the workflow jobs and establishes mappings with the resource information available on the Cloud. The results show that the proposed approaches can capture the Cloud-Aware Provenance (CAP) by capturing the information related to Cloud infrastructure (virtual machines) used during a workflow execution. It can then re-provision a similar execution infrastructure with same resource configurations on the Cloud using CAP to reproduce a workflow execution. Figure 5 shows that the workflow execution time remains the same for reproduced workflow because similar execution infrastructure was provisioned using the Cloud-Aware Provenance. The workflow reproducibility is verified by comparing the outputs produced by the workflows. In this regard, the proposed algorithm (see Algorithm 2) compares the outputs produced by two given workflows. Furthermore, this paper also presents the impact of the devised mapping approaches on the workflow execution time. The result in Figure 6 shows that the presented mapping approaches do not significantly affect the workflow execution time because they work outside the virtual machine and do not interfere with the job execution. In future, the proposed approach will be extended and a detailed evaluation of the ReCAP framework will be conducted. Different performance matrices such as the impact of different resource configuration on workflow execution performance, and total resource provisioning 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. Moreover, the Re-

CAP framework 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.

Acknowledgments. This research work has been funded by a European Union FP-7 project, N4U neuGrid4Users (grant agreement n. 283562, 2011-2014). Besides this, the support provided by OSDC by offering a free Cloud infrastructure of 20 cores is highly appreciated.

References

1. Mehmood, Y., Habib, I., Bloodsworth, P., Anjum, A., Lansdale, T., McClatchey, R.: A middleware agnostic infrastructure for neuro-imaging analysis. In: Computer-Based Medical Systems, 2009. CBMS 2009. 22nd IEEE International Symposium on. (Aug 2009) 1–4
2. Munir, K., Kiani, S.L., Hasham, K., McClatchey, R., Branson, A., Shamdasani, J.: Provision of an integrated data analysis platform for computational neuroscience experiments. Journal of Systems and Information Technology **16**(3) (2014) 150–169
3. Deelman, E., Gannon, D., Shields, M., Taylor, I.: Workflows and e-science: An overview of workflow system features and capabilities. Future Generation Computer Systems **25**(5) (2009) 528 – 540
4. Foster, I., Kesselman, C., eds.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1999)
5. Mell, P.M., Grance, T.: Sp 800-145. the nist definition of cloud computing. Technical report, Gaithersburg, MD, United States (2011)
6. Deelman, E., Singh, G., Livny, M., Berriman, B., Good, J.: The cost of doing science on the cloud: The montage example. In: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing. SC '08, USA, IEEE Press (2008) 50:1–50:12
7. Juve, G., Deelman, E.: Scientific workflows and clouds. Crossroads **16**(3) (March 2010) 14–18
8. Simmhan, Y.L., Plale, B., Gannon, D.: A survey of data provenance in e-science. SIGMOD Rec. **34**(3) (September 2005) 31–36
9. Azarnoosh, S., Rynge, M., Juve, G., Deelman, E., Niec, M., Malawski, M., da Silva, R.: Introducing precip: An api for managing repeatable experiments in the cloud. In: Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on. Volume 2. (Dec 2013) 19–26
10. Belhajjame, K., Roos, M., Garcia-Cuesta, E., Klyne, G., Zhao, J., De Roure, D., Goble, C., Gomez-Perez, J.M., Hettne, K., Garrido, A.: Why workflows break - understanding and combating decay in taverna workflows. In: Proceedings of the 2012 IEEE 8th International Conference on E-Science (e-Science). E-SCIENCE '12, USA, IEEE Computer Society (2012) 1–9
11. Vouk, M.: Cloud computing – issues, research and implementations. In: Information Technology Interfaces, 2008. ITI 2008. 30th International Conference on. (June 2008) 31–40
12. Zhao, Y., Fei, X., Raicu, I., Lu, S.: Opportunities and challenges in running scientific workflows on the cloud. In: Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2011 International Conference on. (Oct 2011) 455–462

13. Shamdasani, J., Branson, A., McClatchey, R.: Towards semantic provenance in cristal. In: Third International Workshop on the role of Semantic Web in Provenance Management (SWPM 2012). (2012)
14. Stevens, R.D., Robinson, A.J., Goble, C.A.: mygrid: personalised bioinformatics on the information grid. Bioinformatics **19** (2003) i302–i304
15. de Oliveira, D., Ogasawara, E., Baiao, F., Mattoso, M.: Scicumulus: A lightweight cloud middleware to explore many task computing paradigm in scientific workflows. In: Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on. (July 2010) 378–385
16. Ko, R., Lee, B., Pearson, S.: Towards achieving accountability, auditability and trust in cloud computing. In Abraham, A., Mauri, J., Buford, J., Suzuki, J., Thampi, S., eds.: Advances in Computing and Communications. Volume 193 of Communications in Computer and Information Science. Springer Berlin Heidelberg (2011) 432–444
17. Foster, I., Vöckler, J., Wilde, M., Zhao, Y.: Chimera: a virtual data system for representing, querying, and automating data derivation. In: Scientific and Statistical Database Management, 2002. Proceedings. 14th International Conference on. (2002) 37–46
18. Scheidegger, C., Koop, D., Santos, E., Vo, H., Callahan, S., Freire, J., Silva, C.: Tackling the provenance challenge one layer at a time. Concurr. Comput. : Pract. Exper. **20**(5) (April 2008) 473–483
19. Kim, J., Deelman, E., Gil, Y., Mehta, G., Ratnakar, V.: Provenance trails in the wings-pegasus system. Concurr. Comput. : Pract. Exper. **20**(5) (April 2008) 587–597
20. Zhang, O.Q., Kirchberg, M., Ko, R.K., Lee, B.S.: How to track your data: The case for cloud computing provenance. In: Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on, IEEE (2011) 446–453
21. 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., Duc, H.: Tracking of data leaving the cloud. 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications **0** (2012) 137–144
22. Macko, P., Chiarini, M., Seltzer, M.: Collecting provenance via the xen hypervisor. 3rd USENIX Workshop on the Theory and Practice of Provenance (TAPP) (2011)
23. Chirigati, F., Shasha, D., Freire, J.: Reprozip: Using provenance to support computational reproducibility. In: Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance. TaPP '13, Berkeley, CA, USA, USENIX Association (2013) 1:1–1:4
24. Janin, Y., Vincent, C., Duraffort, R.: Care, the comprehensive archiver for reproducible execution. In: Proceedings of the 1st ACM SIGPLAN Workshop on Reproducible Research Methodologies and New Publication Models in Computer Engineering. TRUST '14, New York, NY, USA, ACM (2014) 1:1–1:7
25. Santana-Perez, I., Ferreira da Silva, R., Rynge, M., Deelman, E., Pérez-Hernández, M., Corcho, O.: A semantic-based approach to attain reproducibility of computational environments in scientific workflows: A case study. In Lopes, L., Žilinskas, J., Costan, A., Cascella, R., Kecskemeti, G., Jeannot, E., Cannataro, M., Ricci, L., Benkner, S., Petit, S., Scarano, V., Gracia, J., Hunold, S., Scott, S., Lankes, S., Lengauer, C., Carretero, J., Breitbart, J., Alexander, M., eds.: Euro-Par 2014: Parallel Processing Workshops. Volume 8805 of Lecture Notes in Computer Science. Springer International Publishing (2014) 452–463
26. Sandve, G.K., Nekrutenko, A., Taylor, J., Hovig, E.: Ten simple rules for reproducible computational research. PLoS Comput Biol **9**(10) (10 2013) e1003285

27. C., S.V.: Reproducible research: Addressing the need for data and code sharing in computational science. Computing in Science & Engineering **12** (2010)
28. Santana-Perez, I., Ferreira da Silva, R., Rynge, M., Deelman, E., Perez-Hernandez, M.S., Corcho, O.: Leveraging semantics to improve reproducibility in scientific workflows. In: The reproducibility at XSEDE workshop. (2014)
29. Vöckler, J.S., Juve, G., Deelman, E., Rynge, M., Berriman, B.: Experiences using cloud computing for a scientific workflow application. In: Proceedings of the 2Nd International Workshop on Scientific Cloud Computing. ScienceCloud '11, USA, ACM (2011) 15–24
30. Howe, B.: Virtual appliances, cloud computing, and reproducible research. Computing in Science Engineering **14**(4) (July 2012) 36–41
31. Zhao, Y., Li, Y., Raicu, I., Lu, S., Tian, W., Liu, H.: Enabling scalable scientific workflow management in the cloud. Future Generation Computer Systems (0) (2014) –
32. Lifschitz, S., Gomes, L., Rehen, S.K.: Dealing with reusability and reproducibility for scientific workflows. In: Bioinformatics and Biomedicine Workshops (BIBMW), 2011 IEEE International Conference on, IEEE (2011) 625–632
33. Missier, P., Woodman, S., Hiden, H., Watson, P.: Provenance and data differencing for workflow reproducibility analysis. Concurrency and Computation: Practice and Experience (2013)
34. Abrishami, S., Naghibzadeh, M., Epema, D.H.: Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds. Future Generation Computer Systems **29**(1) (2013) 158 – 169 Including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures.
35. Malawski, M., Juve, G., Deelman, E., Nabrzyski, J.: Algorithms for cost- and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds. Future Generation Computer Systems **48**(0) (2015) 1 – 18 Special Section: Business and Industry Specific Cloud.
36. Woodman, S., Hiden, H., Watson, P., Missier, P.: Achieving reproducibility by combining provenance with service and workflow versioning. In: Proceedings of the 6th Workshop on Workflows in Support of Large-scale Science. WORKS '11, USA, ACM (2011) 127–136
37. Groth, P., Deelman, E., Juve, G., Mehta, G., Berriman, B.: Pipeline-centric provenance model. In: Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science. WORKS '09, USA, ACM (2009) 4:1–4:8
38. Horta, F., Silva, V., Costa, F., de Oliveira, D., Ocaña, K., Ogasawara, E., Dias, J., Mattoso, M.: Provenance traces from chiron parallel workflow engine. In: Proceedings of the Joint EDBT/ICDT 2013 Workshops. EDBT '13, New York, NY, USA, ACM (2013) 337–338
39. Tannenbaum, T., Wright, D., Miller, K., Livny, M.: Beowulf cluster computing with linux. MIT Press, Cambridge, MA, USA (2002) 307–350
40. Latchoumy, P., Khader, P.S.A.: Survey on fault tolerance in grid computing. International Journal of Computer Science & Engineering Survey (IJCSES) Vol **2**(2011) (2011)
41. Stallings, W.: Cryptography and Network Security: Principles and Practice. 5th edn. Prentice Hall Press, Upper Saddle River, NJ, USA (2010)
42. Ramakrishnan, L., Plale, B.: 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, USA, ACM (2010) 4:1–4:12