Provenance in the Cloud: *Why* and *How*?

Muhammad Imran
Research Group Entertainment Computing
University of Vienna, Austria
Email: imran.mm7@gmail.com

Helmut Hlavacs
Research Group Entertainment Computing
University of Vienna, Austria
helmut.hlavacs@univie.ac.at

***Abstract*—Provenance is an important aspect in the verification, audit trails, reproducibility, privacy and security, trust, and reliability in many fields ranging from art, food production, medical sciences, in-silico experiments, and distributed computing. On the other hand, Cloud computing is the business model of distributed computing and is considered the next generation of computing and storage platforms. Cloud computing requires an extension of the architecture of distributed and parallel systems by using virtualization techniques. Key to this extensible architecture is to support properties such as compute "on demand" and "pay as you go" model. Clouds are in use since a few years and they already expanded in the business domain (Amazon EC2, Microsoft Azure, IBM SmartCloud) and research environments (EUCALYPTUS, OpenNebula, Nimbus). Many research domains have already adopted Cloud technology into their existing computational and storage platforms and, thus, a shift of technology is in progress.**

In this paper, we present provenance description in computing sciences. Then, we give an overview of Cloud architecture and answer why provenance is important for Cloud computing. We introduce a mechanism to include provenance in the Cloud which requires minimal knowledge and understanding of underlying services and architecture. Therefore, we detail the importance along with the characteristics identified and present a framework for provenance in Cloud computing. We assure trust by augmenting a Cloud infrastructure with provenance collection in a structured way and present first performance results of the extended architecture. Finally, we discuss the results and summarize challenges and open issues of provenance in Clouds.

***Keywords*-provenance; research or open Clouds.**

I. INTRODUCTION

Oxford dictionary [1] defines provenance as "the place of origin or earliest known history of something". In many fields including art, science and computing, provenance is considered as the first class data of importance for tracing an object to its origin. Provenance is defined by a set of different properties about the process, time, and input and manipulated data. Provenance is used to answer a few basic questions such as when the object was created, the purpose of creation, and where the object originated from (e.g., the creator of the object).

In computing sciences, a provenance system is used to collect, parse, and store related metadata. Such data is used for verification and tracking back, assurance of reproducibility, trust, and security, fault detection, and audit trials. These metadata include functional data required to trace back the creation process of objects and results, but also non-functional data such as the performance of each step including, e.g., energy consumption.

Since Cloud is an evolving technology which is based on virtualization and offer, on-demand computing, pay-as-you-go model, and is highly scalable and more abstract. There is a strong need to propose a provenance scheme for this dynamic, abstract and distributed environment. In addition to challenges for distributed computing, the abstraction and highly flexible usage pose new demands, i.e., a provenance framework for Clouds has to support these issues. Rajendra Bose et al. [2] present a detailed survey of computational models and provenance systems in distributed environment, specifically workflows execution. However, none of the approaches support provenance in the Cloud environment. These existing schemes rely on the support of native services from distributed or workflow computing, e.g., process schedulers. Generally, provenance systems in grid, workflow, and distributed computing are either strongly part of the enactment engine or they use Application Programming Interfaces (APIs), which are enactment engine specific [3].

Cloud infrastructure is not extensible by nature and therefore, existing techniques are not a good fit to Cloud environment and to address Cloud specific challenges. A better approach is to follow an independent and modular provenance scheme as described in [4]. Such a scheme is possible by extending the middleware of Cloud infrastructure where various components and services are deployed (extension of third party tools and libraries). This scheme which is a loosely coupled (domain and application independent) works independently of Cloud infrastructure, client tools and is of high importance to support future e-science.

In this paper, we provide a general discussion of provenance in different fields with a particular focus on open or research Clouds. We present underlying architecture of open Cloud, and propose a framework for provenance data collection in the Cloud. Hereby, we address the most important properties of a provenance system that is, independence of the Cloud architecture, low storage and computational overhead of provenance data, and usability. Provenance for Clouds to the best of our knowledge has not been fully addressed yet. The major contributions of this paper are following:

- analysis of provenance in distributed computing, giving reasons of the importance and highlight challenges of provenance in the Clouds and distributed environment;
- a novel proposed scheme which can be deployed to the Cloud environment while addressing different vendors and architectures;
- first performance test results of the provenance framework.

The rest of the paper is organized as follows. In Section II, we discuss the related work in computing sciences. In Section III, Cloud architecture is discussed along with a presentation of Eucalyptus Cloud and its dependencies tools and applications. Section IV presents challenges and provenance data applied to Cloud computing. In Section V, we discuss the proposed framework, configuration of provenance system to Cloud middleware and its main components. Section VI describes first test results and Section VII concludes our work and details future implementation directions.

II. Related Work

Numerous techniques and projects have been proposed during the last few years for provenance systems in computational sciences for validation, reproduction, trust, audit trials and fault tolerance. These techniques range from tightly coupled provenance system to loosely coupled systems [5]–[8]. Provenance Aware Service Oriented Architecture (PASOA) [9], [10] uses Service Oriented Architecture (SOA) [11] for provenance collection and its usage in distributed computing for workflow management systems. myGrid [12] and Kepler [13] are examples of projects for executing in-silico experiments developed as workflows and they use Taverna [14] and Chimera [15] schemes respectively for Provenance data management in these computational systems. However, none of these approaches were designed specifically for Cloud computing architecture. Recently, Muniswamy-Reddy et al. [16] discussed the importance of provenance for Cloud computing services offered by AMAZON EC2 [17] using Provenance-Aware Storage Systems (PASS) [18] system.

In the e-science domain, experiments are performed in dry labs (in-silico); provenance system has to address data collection and availability in distributed environment. Provenance systems use different methods and approaches to address these challenges. Each approach has pros and cons which are related properties of a provenance system in distributed computing. Distributed computing challenges in general and Cloud specific challenges in particular are discussed in detail in Section IV-B, where Section IV-C gives a brief overview of Cloud specific provenance data.

III. Cloud architecture

Cloud vision is to address a complex engineering, medical or social problem by mega scale simulation and handling huge amount of data with a massive computation power. Clouds are generally categorized as business cloud, research or private cloud and hybrid cloud. IaaS (Infrastructure as a Service), PaaS (Platform as a Service) and SaaS (Software as a Service) are the terms heavily used in a Cloud computing paradigm and is mostly broken into these three segments.

IaaS: a service provided for the infrastructure (hardware and software) over the internet. Such an architecture provides servers, virtualized operating systems and data storage units. Elastic Cloud is a commonly used term for IaaS and users pay for required resources as they go. Amazon Elastic Compute Cloud (Amazon EC2), Nimbus [19], OpenNebula [20] and EUCALYPTUS [21] are the leading examples of IaaS. PaaS and SaaS are built on top of IaaS. PaaS provides an interface for software developers to build new or extend existing applications, e.g., Google App Engine and Microsoft Azure. SaaS is an application service provided to the end user by a vendor, e.g., google mail.

Private Cloud IaaS schemes are mostly used in a research environment and small businesses by using open source technologies. They are rapidly growing in the size and magnitude and expanding in different domains. With the new technologies and advancements, a private Cloud can be part of other public or private Clouds thus, providing the functionality of a hybrid Cloud.

A. EUCALYPTUS

Eucalyptus is an open source implementation of Cloud computing IaaS scheme using JAVA and C/C++ for various components. Users can control an entire Virtual Machine (VM) instance deployed on a physical or virtual resource [22]. It supports modularized approach and is compatible with industry standard in Cloud, i.e., Amazon EC2 and its storage service S3. It is one of the most used platforms to create scientific and hybrid Clouds. Eucalyptus gives researchers the opportunity to modify and instrument the software which is been lacking in the business offerings, e.g., Amazon EC2.

Figure 1 presents the extended architecture of Eucalyptus Cloud. There are three main components involved: Cloud Controller (CLC, i.e., middleware), Cluster Controller (CC) and Node Controller (NC). CLC, CC and NC communicates with each other and outside applications using Mule [23] and Apache Axis2/C framework. CLC interacts with CC, where CC is the part of Cloud used to manage clusters in the network. CC interacts and controls different NCs by associating and differentiating them using unique addresses and also balancing load in the cluster. NC assign a VM for the job execution submitted by a user. Walrus is web service used for distributed storage management of virtual images and users metadata. All the communications between different components of Eucalyptus Cloud is achieved by using SOAP, XML, WSDL, and HTTP communication protocols via Axis2/C and Mule framework.

IV. Provenance in Cloud: *Why*

There are various definitions of Cloud computing (utility computing, autonomic computing) and is used as per the understandings, knowledge and requirements by different companies and users. Yes, there are some differences from previous computing technologies specifically to mention virtualization, on demand, pay as you go model, extremely flexible and more abstract. Ian Foster et al. [24] present an overview of the major differences between Cloud and grid and mentions the most important feature of Cloud technology is the total dependence on services (SOA architecture). There is underlying architecture for networking of software and hardware but, to the end user it is completely abstract and hidden. The abstraction allows the end user to send data to Cloud and get data back, without bothering about the underlying details. This behavior is fine for a normal user but, in research environment, scientists are more interested in the overall process of execution and a step by step information to keep a log of sub-data and sub-processes to make their experiments believable, trust able, reproducible and to get inside knowledge. With improvements of in-silico experiments, most of the computation and processing is done by using computing resources and not in a real lab.

Users of Cloud environment may not be interested in the physical resources, e.g., brand of computer but, surely they are interested in the invoked service, input and output parameters, time stamps of invocation and completion, overall time used by a process, methods invoked inside a service and the overall process from start to finish. This metadata which provides the user an ability to see a process from start to end or simply track back to find the origin of a final result is called provenance. Generally provenance is used in different domains by scientists and researchers to trust, track back, verify individual input and output parameters to services, sub process information, reproducibility, compare results and change preferences (parameters) for another simulation run. Provenance is still missing in Cloud environment and needs to be explored in detail as mentioned in [16], [25].

A. *Implication of a Provenance Enabled Cloud*

Introducing the provenance data into Cloud infrastructure would result in following advantages:

- Patterns: The use of provenance data to find patterns in the Cloud resources usage. These patterns can be further utilized to forecast a future request.
- Trust, reliability and data quality: The final data output can be verified based on the source data and transformation applied.
- Resources utilization: In Cloud, provenance data can be used to utilize the existing running resources by allocating copy of a running resource. This will be achieved by comparing a new request to the already running resources and this information is available in provenance data.

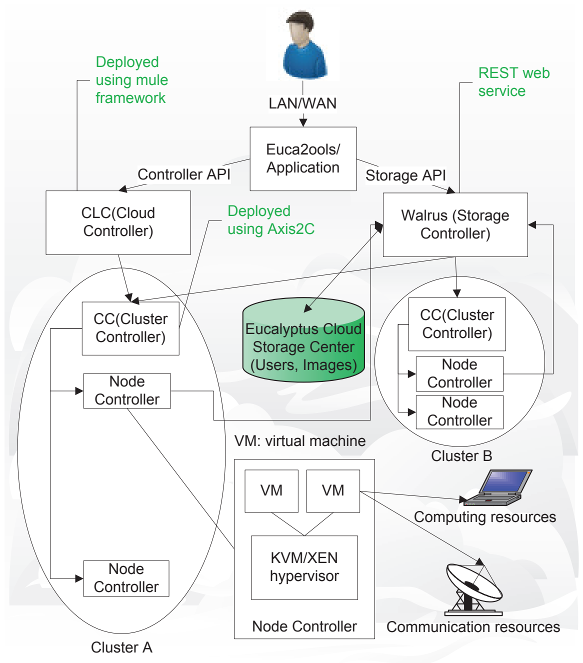

Figure 1. Extended architecture of Eucalyptus Cloud.

- Reduced cost and energy consumption: Provenance data results in a cost and energy efficiency by using patterns to forecast a future request and by utilizing existing running resources.
- Fault detection: Provenance data can pinpoint the exact time, service, method and related data in case of a fault.

B. *Provenance Challenges in Cloud*

Usual provenance challenges include: collecting provenance data in a seamless way with a modularized design and approach, with minimal overhead to object identification, provenance confidentiality and reliability, storing provenance data in a way so it can be used more efficiently (energy consumption) and presenting such information to the end user (query, visualization). Cloud brings more challenging to these existing challenges because we have to address the scalable, abstract and on demand architecture of Cloud. A provenance system in Cloud should address the following challenges:

- Domain, Platform and Application independence: How the provenance system works with different domain (scientific, business, database), platforms (windows, linux) and applications.
- Computation overhead: How much extra computation overhead is required for a provenance system in a particular domain.
- Storage overhead: How and where is the provenance data stored. It depends on the type, i.e., copy of original data or a link reference to original data, granularity

(coarse-grained or fine-grained) and storage unit (SQL server, mySQL, file system) of provenance data.

- Usability: It determines the ease of use of a provenance system from a user and Cloud resources provider perspective. How to activate, deactivate and embed a provenance system into existing Cloud infrastructure and services, e.g., is it completely independent or modification is required on Cloud services layer.
- Object identification: Identify an object in the Cloud and link the provenance data to source by keeping a reference or by making a copy of the source object.
- Automaticity: With huge amount of data and process computation within Cloud, collecting and storing provenance data should be automatic and consistent.
- Cloud architecture: Addressing the on-demand, abstract and scalable structure of Cloud environment with availability and extensibility of different components.
- Interaction with Cloud services: Cloud services are not extensible therefore, they cannot be modified. Business Clouds are propriety of organizations and open source Clouds needs understanding of every service if change is required. The better approach is to provide an independent provenance scheme which requires no change in the existing services architecture.

C. *Provenance Data*

A provenance system should address two different perspectives in collecting metadata for Cloud architecture. Applications running on Cloud as SaaS or PaaS and provenance of Cloud infrastructure (IaaS). Users of Cloud are more interested in their application provenance where, providers are interested in IaaS services provenance to observe resource usage and find patterns in applications submitted by users to provide with a more sophisticated model for resources usage. Following, is the list of mandatory metadata in a Cloud environment:

1) Cloud process data: Cloud code execution and control flow between different processes (web services), e.g., in EUCALYPTUS are CLC, CC and NC services. Web service and method name in particular.
2) Cloud data provenance: Data flow, input and output datasets which are consumed and produced and parameters passing between different services.
3) System provenance: System information or physical resources details, e.g., compiler version, operating system and the location of virtualized resources.
4) Timestamps: Invocation and completion time of Cloud services and methods.
5) Provider and user: Details about Cloud users and services provider, e.g., location of clusters and nodes. Different providers have different trust level and there could be laws against usage of resources for a particular geographical area.

V. PROVENANCE FRAMEWORK: *How*

A Cloud infrastructure is deployed and it relies on the open source third party tools, libraries and applications. Eucalyptus Cloud in particular depends on the Apache Axis, Axis2/C, and Mule framework. These third party libraries are used for the communication mechanism between various components of Cloud infrastructure. Cloud infrastructure is the orchestration of different services and the third party libraries works as a middleware to connect these services. The purpose of Cloud computing is more abstraction than previous technologies like Grid and Workflow computing and therefore, Cloud services are not extensible.

One method to implement provenance into the Cloud infrastructure is by changing the source code. This could be very cumbersome as deep understanding of the code is required. This will also restrict the change to the particular version of the Cloud. This method is not feasible to address the provenance challenge for various Cloud providers, domains and applications. The second method is to capture the provenance data on the middleware of a Cloud. This is possible by extending the third party libraries used by Cloud infrastructure and add custom methods to collect provenance data at various different levels. Such a scheme will lead to the minimum efforts and can be deployed across any Cloud scheme. Further, there will be no change required in Cloud services architecture or signature. To understand this techniques and hence the proposed provenance framework, we give a brief overview to the most important Mule and Axis2/C architecture.

A. *Mule Enterprise Service Bus*

Mule is a lightweight Enterprise Service Bus (ESB) written in JAVA and is based on Service Oriented Architecture (SOA). Mule enables the integration of different application regardless of the communication protocol used by those applications. Eucalyptus CLC services are deployed using Mule framework. CLC services are divided into different components including core, cloud, cluster manager, msgs, etc. These different components are built and deployed as jar files and they use Mule framework messaging protocols (HTTP, SOAP, XML, etc.) to communicate with each other and with other Eucalyptus services (NC and CC).

Extending Mule: Mule framework is based on layered architecture and modular design. Mule offers different kind of interceptors (EnvelopeInterceptor, TimeInterceptor and Interceptor) to intercept and edit the message flow. Since, provenance is metadata information flowing between different components (services) and we do not need to edit the message structure; therefore, we use EnvelopeInterceptor. Envelop interceptors carries the message and are executed before and after a service is invoked.

Configuring Mule Interceptor: There are two steps involved for configuring a Mule interceptors to Cloud services. First step is to built a provenance package (JAVA class files) and copying to the Cloud services directory. Second step requires editing Mule configuration files used by different CLC components. Interceptors can be configured globally to a particular service or locally to a particular method of a service. Listing 1 is a sample “eucalyptus-userdata.xml” mule configuration file used to verify user credentials and groups.

Listing 1. Configuration of Provenance into Mule

```
<?xml version="1.0" encoding="UTF-8"?>
<mule xmlns="http://www.mulesource.org/...">
  <interceptor-stack name="CLCProvenance">
    <custom-interceptor class="eucalyptus.CLC
                        provenance"/>
!.. indicating path of the package and class name
 for CLC services provenance data
  </interceptor-stack>
  <model name="eucalyptus-userdata">
    <service name="KeyPair">
      <inbound>
        <inbound-endpoint ref="KeyPairWS"/>
      </inbound>
      <component>
        <interceptor-stack ref="CLCProvenance"/>
!.. configuring "keypair service" to provenance
 module
        <class="com.eucalyptus.keys.KeyPair
                                Manager"/>
      </component>
      <outbound>
        <outbound-pass-through-router>
          <outbound-endpoint ref="ReplyQueue
                                Endpoint"/>
        </outbound-pass-through-router>
      </outbound>
    </service>
  </model>
</mule>
```

B. Axis2/C Architecture

Eucalyptus NC and CC services are exposed to other components by using Apache Axis2/C framework. Axis2/C is extensible by using handlers and modules [26]. Handlers are the smallest execution unit in Apache engine and are used for different purposes, e.g., web services addressing [27] and security [28]. A message flow between different components of CC and NC go through Axis2/C engine and we deploy custom handlers for provenance data collection inside Axis2/C. Similar concept is used in [29] for workflow services deployed in a tomcat container. This framework is not extensible to Cloud services and architecture. We differ from that work in many factors including interceptors for Mule, Apache Axis and Apache Axis2/C. There is no tomcat container available for Cloud services to deploy the provenance framework and Cloud services use HTTP, XML, SOAP and REST based protocols. Further, Our framework is developed for Cloud services provenance data collection and therefore, parsing, storing, and accessing provenance data is different than their architecture.

Configuration: Axis2/C modules and handlers can be configured globally to all services by editing axis2.xml file, or to a particular service and method by modifying servies.xml file. Listing 2 describes the configuration of provenance module to Eucalyptus NC service.

Listing 2. Configuration of Provenance into Axis2/C

```
<?xml version="1.0" encoding="UTF-8"?>
<service name = "EucalyptusNC">
  <module ref="NCprovenance"/>
!..this will configure provenance to all methods
 in NC
  <Operation name="ncRunInstance">
    <Parameter name = "wsmapping">
      EucalyptusNCncRunInstance
    </Parameter>
  </Operation>
  <Operation name="ncAttachVolume">
    <module ref="NCprovenance"/>
!..this will configure provenance to this
 particular method
    <Parameter name = "wsmapping">
      EucalyptusNCncAttachVolume
    </Parameter>
  </Operation>
</Service>
```

C. Framework Components

Proposed framework is divided into the following components to address the modularity and layered architecture:

- Provenance collection: Collecting important provenance data in a seamless and modular fashion using Mule, Apache Axis and Axis2/C interceptors.
- Provenance storage: Provenance data can be stored as part of Cloud storage unit or, to a dedicated database system. Properly indexing and linking provenance data to original data objects is compulsory.
- Provenance query and visualization: Providing an interface to query provenance data and visualize the results in a graph or chart form.
- Provenance usage: Using collected provenance data to enhance the trust on Cloud environments, reproducibility of applications and fault detection etc. Provenance usage is the extension of provenance query by providing with a standard output to make it compatible with other systems. Particular usage of provenance data is to find access pattern in resources usage, resources utilization (energy consumption) and faults detection.

Figure 2 describes the extended architecture of Axis2/C (particular version of Apache Axis used by EUCALYPTUS) and Mule, the main components of proposed framework and the deployment of provenance module.

VI. Testing and Evaluation

Test cases are performed on Mule and Axis2/C framework with provenance module for collection, parsing and logging metadata. Here, we present results for time increase with provenance module in Axis2/C for the execution chains called Inflow and Outflow. The underlying architecture and system details are following:

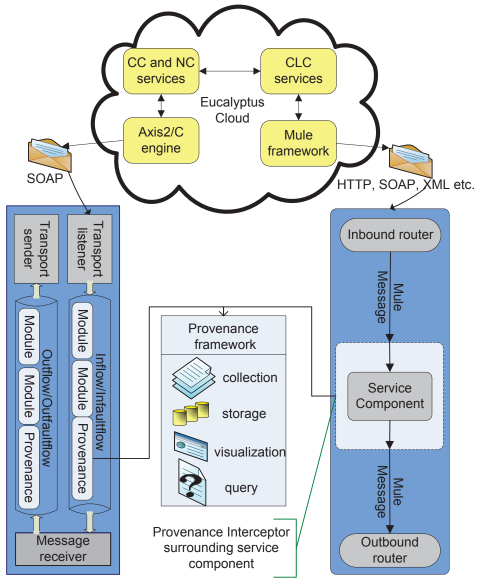

Figure 2. Framework components.

Operating system: Ubuntu 10.04, Processor: Intel Core 2 (2 GHz), RAM: 2 GB, Axis2/C version: 1.6.0, Web service: Echo

Echo service is invoked 100 times in a row for getting real data for comparison. Five multiple runs are performed for the calculation of best time, worst time and average time of execution. The process is executed by considering overall (Inflow and Outflow), only Inflow and only Outflow provenance. Apache Axis2/C engine is extended by using custom handlers and modules in the corresponding flows.

Figure 3 presents the performance of these different execution runs on Axis2/C engine. Left side of the figure details multiple runs of echo service without provenance, with provenance (Inflow and Outflow), only Inflow and only Outflow provenance. Y-axis represents the time required for execution. Right side of the figure shows the increase in time by comparison to without provenance. This increase in time is calculated for overall provenance, only Inflow provenance and only outflow provenance. The comparison is done for average values by using formula 1, where T_2 is time including provenance and T_1 is time excluding provenance.

$$Time\,increase = T_2 - T_1 \tag{1}$$

The average increase in time for 100 simulation runs of echo service for collecting and logging overall provenance data is only 0.017 ms when compared to the execution without provenance. The average increase in time for only Inflow provenance is 0.009 ms and for only Outflow provenance is 0.013 ms when compared to without provenance. The individual Inflow and Outflow provenance was collected for experimental purposes to observe the respective overhead. In a real lab experiment the overall provenance of process is essential. The increase in time is too less and negligible when considering the advantages like fault tracking, resource utilization, patterns finding and energy consumption of a provenance enabled Cloud. Furthermore, the successful deployment of provenance collection to Axis2/C and Mule frameworks support our theory of a generalized and independent provenance framework

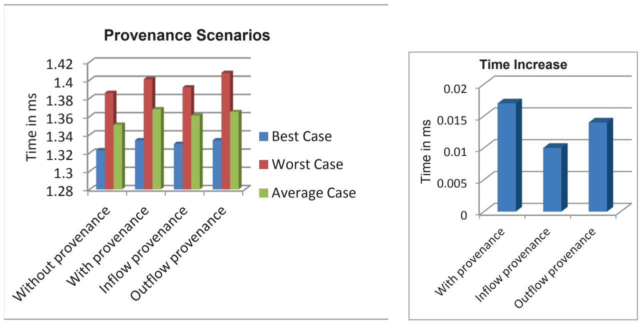

Figure 3. Test results of Echo service.

VII. Conclusion and Future Work

Provenance is an important aspect in e-science. With the technology shift and changes, open Clouds are becoming an important part of e-science. Open Clouds are used in research and private business domain for storage, computation and execution of complex scientific applications. This paper considers provenance as an important metadata for Cloud environment and present provenance properties, Cloud architecture, open Clouds dependencies, and finally propose a framework. Proposed framework can be deployed to any Cloud scheme without modifying the basic services architecture or source code. Further, we gave a brief overview for the need of provenance in Cloud and present the major challenges and properties of such a framework. An independent system is proposed with advantages being simple, easy to use, easy to deploy, and works with open Cloud providers.

In future work, we will give insight details of the framework, simple user interface to configure provenance to Cloud service, evaluation of provenance framework for Cloud services and working example of provenance usage for fault detection and resources utilization (energy consumption).

References

[1] Oxford dictionaries. [retrieved: may, 2012]. [Online]. Available: http://oxforddictionaries.com/definition/provenance

[2] R. Bose and J. Frew, "Lineage retrieval for scientific data processing: a survey," *ACM Comput. Surv.*, vol. 37, no. 1, pp. 1–28, Mar. 2005.

[3] Y. L. Simmhan, B. Plale, and D. Gannon, "A Survey of Data Provenance Techniques," Computer Science Department, Indiana University, Bloomington IN, Tech. Rep., 2005.

[4] A. Marinho, L. Murta, C. Werner, V. Braganholo, S. M. S. d. Cruz, E. Ogasawara, and M. Mattoso, "Provmanager: a provenance management system for scientific workflows," *Concurrency and Computation: Practice and Experience*, 2011.

[5] M. Szomszor and L. Moreau, "Recording and reasoning over data provenance in web and grid services." in *CoopIS/DOA/ODBASE*, ser. Lecture Notes in Computer Science, R. Meersman, Z. Tari, and D. C. Schmidt, Eds., vol. 2888. Springer, 2003, pp. 603–620.

[6] Y. Cui and J. Widom, "Lineage tracing for general data warehouse transformations," in *Proceedings of the 27th International Conference on Very Large Data Bases*, ser. VLDB '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 471–480.

[7] P. Buneman, S. Khanna, and W. chiew Tan, "Why and where: A characterization of data provenance," in *ICDT '01: Proceedings of the 8th International Conference on Database Theory*. Springer, 2001, pp. 316–330.

[8] M. Imran and K. A. Hummel, "On using provenance data to increase the reliability of ubiquitous computing environments," in *Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*, ser. iiWAS '08. New York, NY, USA: ACM, 2008, pp. 547–550.

[9] S. Miles, P. Groth, M. Branco, and L. Moreau, "The requirements of recording and using provenance in e-Science experiments," University of Southampton, Tech. Rep., 2005.

[10] pasoa. [retrieved: may, 2012]. [Online]. Available: http://www.pasoa.org/

[11] oasis. [retrieved: may, 2012]. [Online]. Available: http://www.oasis-open.org/

[12] mygrid project. [retrieved: may, 2012]. [Online]. Available: http://www.mygrid.org.uk/

[13] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. Mock, "Kepler: an extensible system for design and execution of scientific workflows," in *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*, Jun. 2004, pp. 423–424.

[14] Taverna workflow management system. [retrieved: may, 2012]. [Online]. Available: http://www.taverna.org.uk/

[15] I. Altintas, O. Barney, and E. Jaeger-frank, "Provenance collection support in the kepler scientific workflow system," in *In Proceedings of the International Provenance and Annotation Workshop (IPAW)*. Springer-Verlag, 2006, pp. 118–132.

[16] K.-K. Muniswamy-Reddy, P. Macko, and M. Seltzer, "Making a cloud provenance-aware," in *First workshop on on Theory and practice of provenance*, ser. TAPP'09. Berkeley, CA, USA: USENIX Association, 2009, pp. 12:1–12:10.

[17] Amazon elastic compute cloud. [retrieved: may, 2012]. [Online]. Available: http://aws.amazon.com/ec2/

[18] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. I. Seltzer, "Provenance-aware storage systems." in *USENIX Annual Technical Conference, General Track*. USENIX, 2006, pp. 43–56.

[19] Nimbus. [retrieved: may, 2012]. [Online]. Available: http://www.nimbusproject.org/

[20] Opennebula. [retrieved: may, 2012]. [Online]. Available: http://opennebula.org/

[21] Eucalyptus. [retrieved: may, 2012]. [Online]. Available: http://open.eucalyptus.com/

[22] S. Wardley, E. Goyer, and N. Barcet, "Ubuntu enterprise cloud architecture," *Technical White Paper*, Aug. 2009.

[23] Mule esb. [retrieved: may, 2012]. [Online]. Available: http://www.mulesoft.org/what-mule-esb

[24] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared," in *2008 Grid Computing Environments Workshop*. IEEE, Nov. 2008, pp. 1–10.

[25] M. A. Sakka, B. Defude, and J. Tellez, "Document provenance in the cloud: constraints and challenges," in *Proceedings of the 16th EUNICE/IFIP WG 6.6 conference on Networked services and applications: engineering, control and management*, ser. EUNICE'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 107–117.

[26] A. S. Foundation, "Apache axis2/java - next generation web services," Website http://ws.apache.org/axis2/, 2009.

[27] Axis2- ws-addressing implementation. [retrieved: may, 2012]. [Online]. Available: http://axis.apache.org/axis2/java/core /modules/addressing/index.html

[28] Apache axis2/c manual. [retrieved: may, 2012]. [Online]. Available: http://axis.apache.org/axis2/c/rampart /docs/rampartc_ manual.html

[29] F. A. Khan, S. Hussain, I. Janciak, and P. Brezany, "Enactment engine independent provenance recording for e-science infrastructures." in *Proceedings of the Fourth IEEE International Conference on Research Challenges in Information Science RCIS'10*, 2010, pp. 619–630.