Combining Cloud and Grid with a User Interface

Jiaqi Zhao[1], Jie Tao[2], Mathias Stuempert[2], and Moritz Post[3]

[1] School of Basic Science, Changchun University of Technology, P.R. China
[2] Steinbuch Center for Computing, Karlsruhe Institute of Technology, Germany
jie.tao@iwr.fzk.de
[3] Innoopract Informationssysteme GmbH, Karlsruhe, Germany

Abstract. Increasing computing clouds are delivered to customers. Each cloud, however, provides an individual, non-standard user interface. The difference in cloud interfaces must burden the users when they work with several clouds for acquiring the services with expected price. This paper introduces an integrated framework that can be used by cloud users to access the underlying services in a uniform, cloud-independent manner. The framework is an extention of a graphical grid user interface developed within the g-Eclipse project. The goal of building a cloud user interface on top of a grid interface is to combine clouds and grids into a single realm, allowing an easy interoperation between the two infrastructures.

1 Introduction

Since Amazon announced its computing cloud EC2 [1] and storage cloud S3 [2], cloud computing becomes a hot topic. As a consequence, a number of cloud infrastructures have been established, both for commercial and research purpose. Examples are Google App Engine [13], Microsoft Live Mesh [15], Nymbus[8], Cumlus [16], Eucalyptus [5], and OpenNybula [4]. Currently, most of the cloud projects focus on Infrastructure as a Service and Software as a Service, but we are sure that other topics, such as Software as Platform as a Service and HPC as a Service, will be addressed in the near future.

Actually, cloud computing is not a completely new concept. It has similar features with grid computing. A detailed comparison between these two paradigms can be found in [11]. Grid computing has been investigated for thirty years. Many grid infrastructures, especially those at the international level, were well established. Hence, cloud computing will not replace grid computing; rather it provides the user community with additional computing platforms.

Grid computing has ever faced a problem: different middlewares have own requirement for accessing the infrastructure. This problem was solved by building an abstract layer to hide the middleware-specific implementation [7,14]. Cloud computing has the same problems. Currently, each cloud offers a different user interface, mostly command-line, requiring the user to install their client software and learn how to use the commands to request the services.

Our solution is an integrated, intuitive platform that can be used as a generic, standard interface to access any cloud. Users see an identical view, no matter which cloud is accessed. Furthermore, the interface uses graphical presentation, which is easier to operate than command-line options. Besides serving as a cloud interface, the platform

D.R. Avresky et al. (Eds.): Cloudcomp 2009, LNICST 34, pp. 103–111, 2010.

is also a bridge to connect the cloud with the grid. In this case, we build the cloud user interface on top of an existing grid framework that was developed within the g-Eclipse project.

g-Eclipse [7,10] aims at providing a generic framework that allows users to access the power of the existing grid infrastructures via a standardized, customizable, and intuitive interface. This framework is designed for all grid users, operators, and application developers. Grid users can interact with grid resources in a simple, graphical way without having to know the technical details. For example, files can be transferred across grid sites by drag&drop; job submission needs only a mouse click. Resource providers can use the intuitive tools to operate and maintain the grid sites, manage the virtual organizations, and perform benchmarking. Application developers reduce the development cycle with the g-Eclipse support of remote building and deployment tools.

g-Eclipse is designed to support users of various virtual organizations. It uses a layered infrastructure with middleware-independent interfaces and middleware specific functionalities. Currently, standard middleware functionalities are provided.

This work extends g-eclipse with a cloud-independent infrastructure, including editors and views for service presentation and templates for supporting cloud programming models. Based on this infrastructure, various cloud platforms can be connected to the g-Eclipse framework with an individual implementation for accessing the specific cloud. This paper describes the design of the cloud infrastructure and the connection to the Amazon EC2 as an example.

The remainder of the paper is organized as following. Section 1 first gives an introduction to the g-Eclipse framework. This is followed by the concept and design of an integrated cloud user interface in Section 3. Section 4 describes our initial implementation of the proposed concept with EC2 and demonstrates how to access this cloud via the extended g-Eclipse framework. The paper concludes in Section 5 with a brief summary and several future directions.

2 g-Eclipse: Building a Framework to Access the Power of the Grid

The g-Eclipse framework was originally designed to provide a high-level abstraction for accessing grid infrastructures based on traditional grid middleware systems such as gLite [6] or GRIA [9]. It is build on top of the well-known Eclipse framework [12] and makes extensive use of its design-patterns. The abstraction layer – called the Grid Model – unifies the structure and functionality of grids in a set of well defined Java interfaces. Basic implementations of these interfaces for generic functionalities, as well as a UI layer, are provided to present and access underlying infrastructures in a standardized way. On top of these core parts, middleware specific implementations of the Grid Model can be plugged-in. This so called implementation layer enables the access to infrastructures based on the corresponding middlewares.

So far the g-Eclipse project has integrated two different middlewares, i. e. gLite which focuses on the scientific user and GRIA which targets industry and commerce. The current gLite implementation covers all use-cases foreseen in the Grid Model. Therefore, this part may be seen as finalized. The GRIA implementation is in an early

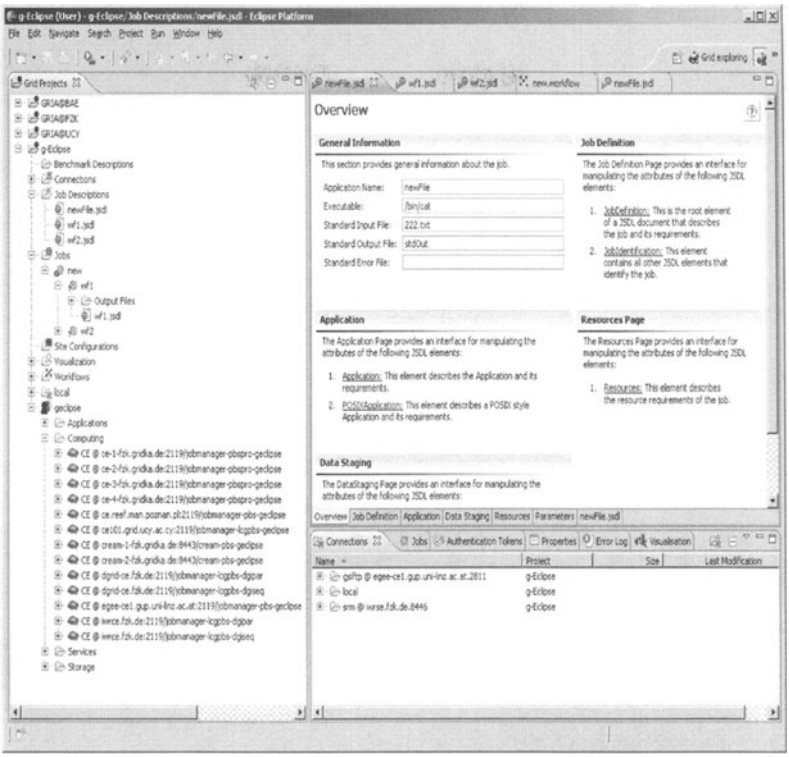

Fig. 1. Screenshot of the g-Eclipse user interface

state and mainly covers the grid user's use cases. Further development, e.g. for Globus Toolkit 4, is currently ongoing.

Fig. 1 shows a screenshot of the g-Eclipse framework for grid users. The left column of the platform is a Grid Project view where all projects created by the user are depicted. Project is a fundamental concept in g-Eclipse. It is the interface for any grid operations. Hence, a project has to be created before any action can be invoked.

The concrete view in Fig. 1 contains four projects where the last one, with the name of g-Eclipse, was expended. Each project consists of several folders for storing temporal files and for presenting information. For example, all established grid connections can be found in the folder "Connection". The lower window on the right side of Fig. 1 depicts the contents of this folder. The three connections are built for different machines to transfer data. Files can be moved from one machine to another by drag&drop. The folder "Job Description" holds all job descriptions that define computing tasks. A job description file can be simply created using a multi-editor shown on the upper side of the right column of Fig. 1. Users need only specify the executables and parameters, a job description with the grid standard is created automatically. The jobs can be submitted with a mouse click and the results are demonstrated in the folder "Jobs". The last folder is a specific one showing the VO related information, including the

deployed applications, the computing and storage resources, as well as the available services.

Overall, g-Eclipse built a platform allowing an easy access to grid infrastructures. It also integrated tools for support application development. More importantly, it enables an interoperation between different grids. Therefore, we select this platform as the base for an intuitive, unified cloud user interface.

3 A Cloud Framework Based on g-Eclipse

We intend to develop a cloud user interface like g-Eclipse for the grid. The interface provides basiç functionalities for accessing a scientific cloud. This includes facilities for authority and authentication, for data management, for service deployment, and for accesses to the computing resources and services. It also contains tools for debugging and visualizing applications, for benchmarking, and for resource management.

Following the g-Eclipse architecture, the cloud interface contains a core and a cloud-specific implementation, where the core plug-ins provide the basic functionality to access a cloud platform. For this, an extension of the g-Eclipse core is essential to define interfaces for cloud specific functionality, e.g. cloud services.

In a cloud world, everything is observed as a service: hardware is a service, software is a service, and infrastructure is a service. Therefore, a cloud access interface must support the presentation, request, and deployment of services. The following components are required:

- A multi-layer editor for users to specify service request.
  A cloud service is combined with various parameters. Different services have also individual formation of the parameters. For example, CPU frequency and memory size are typical specifications for a hardware service, while version number and file size are parameters to describe a software package. The multi-layer editor allows the user to describe the requested services in detail.
- A view for showing the available services.
  The service view will be designed and implemented for presenting the services which are available in a cloud or requested by the users.
- An editor for service deployment and publication.
  Cloud developers or resource providers need an interface to describe new services and then publish them. Again, service related metrics and SLA values are necessary parameters. An editor will be provided for this task.

In addition, cloud computing has its own programming languages and models. Currently, MapReduce [3] is regarded as an adequate paradigm for writing cloud applications. It can be expected that more models will be designed in the future. We intend to develop templates to support application developers, with an initial implementation for MapReduce.

The functionalities listed above are common for all clouds. They form the base for accessing any cloud with g-Eclipse. Additionally, a specific implementation is required for each different cloud to cover its individual feature, in the same way that g-Eclipse handles different grid middlewares. The development work is currently on-going.

4 An Initial Implementation: Access the Amazon Web Service

For verifying our concept of building a cloud framework using g-Eclipse, we first extended this grid user interface with several cloud related components with respect to the Amazon Web Services. We then implemented additional plug-ins for accessing EC2. Theses plug-ins are responsible for handling AWS specific issues, such as accounting, running machine image, and logging in a machine.

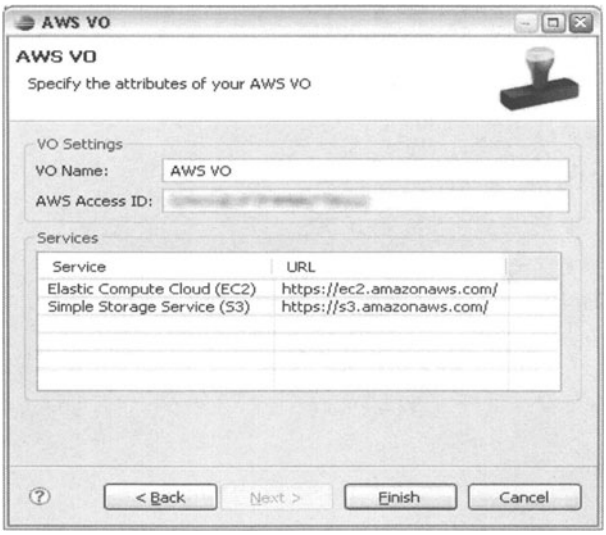

Fig. 2. Screenshot of the VO wizard for creating an AWS VO

In the grid world, any user must be a member of a virtual organization (VO). To access a grid infrastructure with g-Eclipse, a VO has to be created or imported. Cloud computing does not apply the VO concept. However, we make use of this concept in the AWS implementation in order to specify the endpoints for accessing an underlying cloud infrastructure. A screenshot of the wizard for creating an AWS VO is shown in Fig. 2.

As shown in the figure, the VO wizard allows users to define a VO which can be later used to create a project. The user has to specify the name of the VO, the AWS access identifier, and the access points to the underlying clouds. This wizard is implemented for AWS, but can be directly applied for accessing other similar clouds, for example, Eucalyptus.

To use the Amazon cloud services, a user has to provide a secrete ID. This issue is solved in g-Eclipse by reusing its Authentication Token that is actually designed for grid authentication.

Fig. 3 shows the wizard for creating such a token. As can be seen, users can give their AWS credentials using this wizard. g-Eclipse then uses these credentials to create tokens and relies on the tokens to interact with the cloud for authentication.

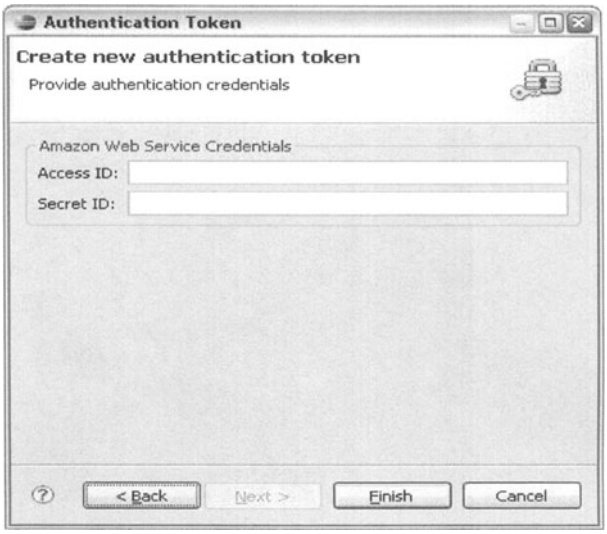

Fig. 3. Screenshot of the wizard for creating an authentication token

As mentioned in Section 1, g-Eclipse uses the project concept for grid actions. This concept is reused for cloud operations. By creating a project bound to a cloud VO a user is able to query and access his personalized resources that are available from the specified cloud services.

Fig. 4 is a screenshot of an AWS project on g-Eclipse, where the VO folder is expanded. It can be seen that the cloud resources, like the Amazon machine images (AMI), are presented in the service subtree of the project's VO. These AMIs are listed in separate folders for distinguishing those owned by the user and those accessible to the user. Furthermore, the user's security groups can be managed within this tree.

From context menu actions a user is able to start instances of these AMIs by creating an Eclipse launch configuration. Fig. 5 shows the corresponding launch dialog that allows the user to specify various parameters such as the type and the number of instances to be launched. In addition, from this dialog it is possible to specify a payload file that is uploaded and made accessible to the running instances. This file is usually used to parameterize these instances. Once one or more instances have been launched they appear in the VO tree as computing nodes. These nodes may be accessed by using the integrated SSH console that is part of g-Eclipse.

After an instance is launched g-Eclipse offers the possibility to access this machine via a SSH shell. In order to use this connection method, the security group used to launch the AMI has to open the port 22 (ssh default port). Because the ssh connection method uses the Eclipse connection infrastructure, the ssh private key has to be inserted into the list of available keys. The running instances can be connected using an action in the context menu. This action opens the SSH login data dialog with the correct external DNS name inserted. The only parameter to be provided is the login name which is "root". There is no need for a password, since it is contained within the ssh private key. Fig. 6 shows a sample dialog.

Fig. 4. Screenshot of a project view with EC2

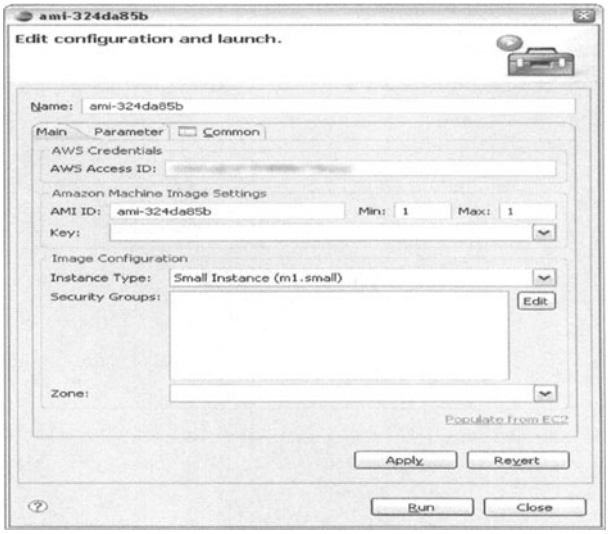

Fig. 5. Screenshot of the launch configuration dialog for launching an AWS machine image

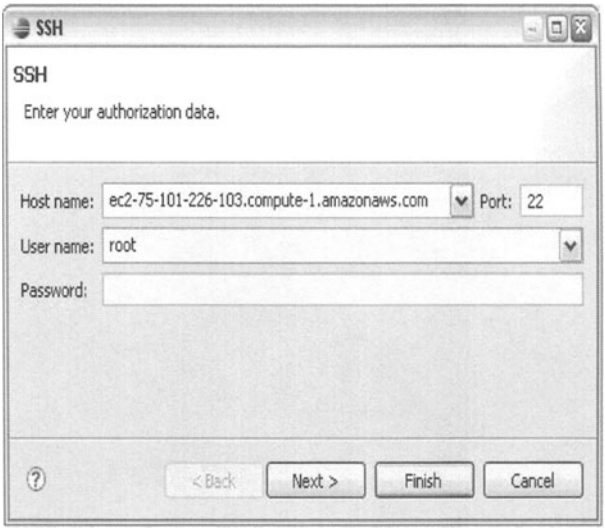

Fig. 6. Screenshot of the SSH login wizard

The Amazon S3 service is integrated as an Eclipse File System implementation. The VO subtree of an AWS project lists the corresponding buckets as storage. From these storage items a user is able to mount these buckets as connections. Such connections appear afterwards in the Connections folder of the project and may be accessed within the project, in the same way of accessing any other folder (local or remote). Files located in these connections may be copied across different connections or just opened, edited and saved on the fly. The underlying g-Eclipse layer for managing EFS implementations ensures interoperability between all available EFS implementations. Therefore, file transfers between S3 and any other EFS implementation is straight-forward.

Overall, we have made it possible to access the Amazon clouds using g-Eclipse with a slight extension of its core architecture and a specific implementation for AWS. This achievement allows the user not only to access the clouds in an easier way but also to move their data across grids and clouds.

5 Conclusion

Cloud platforms are emerging. Different clouds also offer different client side interfaces that are mainly based on command-line designs. To hide the details of cloud client implementations, a generic user interface is required.

This work aims at developing such an interface to both allow cloud users to access the underlying infrastructures in a unified, graphical way and build a bridge between grid and cloud. The interface is an extension of an existing grid framework developed within the g-Eclipse project. To verify our concept, an initial implementation with respect to the Amazon Web Services has been completed. Currently, the entire cloud infrastructure is under development. Furthermore, implementations for connecting other cloud are also planned.

References

1. Amazon Web Services. Amazon Elastic Compute Cloud (Amazon EC2), http://aws.amazon.com/ec2/
2. Amazon Web Services. Amazon Simple Storage Service (Amazon S3), http://aws.amazon.com/s3/
3. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. Communications of the ACM 51(1), 107–113 (2008)
4. Sotomayor, B., et al.: Capacity Leasing in Cloud Systems using the OpenNebula Engine. In: Proceedings of CCA 2008 (2008)
5. Nurmi, D., et al.: The Eucalyptus Open-source Cloud Computing System. In: Proceedings of CCA 2008 (2008)
6. Laure, E., et al.: Programming the Grid with gLite. Computational Methods in Science and Technology 12(1), 33–45 (2006)
7. Kornmayer, H., et al.: gEclipse- An Integrated, Grid Enabled Workbench Tool for Grid Application Users, Grid Developers and Grid Operators based on the Eclipse Platform. In: Proceedings of the 2nd Austrian Grid Symposium, Innsbruck, Austria (September 2006), http://www.geclipse.eu/
8. Keahey, K., et al.: Science Clouds: Early Experiences in Cloud Computing for Scientific Applications. In: Proceedings of CCA 2008 (2008)
9. Surridge, M., et al.: Experiences with GRIA - Industrial applications on a Web Services Grid. In: E-SCIENCE 2005: Proceedings of the First International Conference on e-Science and Grid Computing, pp. 98–105 (2005)
10. Wolniewicz, P., et al.: Accessing Grid computing resources with g-Eclipse platform. Computational Methods in Science and Technologie 13(2), 131–141 (2007)
11. Foster, I.T., Zhao, Y., Raicu, I., Lu, S.: Cloud Computing and Grid Computing 360-Degree Compared. In: Grid Computing Environments Workshop, pp. 1–10 (2008)
12. Gamma, E., Beck, K.: Contributing To Eclipse: Principles, Patterns, And Plug-Ins. Addison-Wesley Professional, Reading (2003)
13. Google. Google App Engine, http://code.google.com/intl/de-DE/appengine/
14. Malawski, M., Bartyński, T., Bubak, M.: A Tool for Building Collaborative Applications by Invocation of Grid Operations. In: Bubak, M., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2008, Part III. LNCS, vol. 5103, pp. 243–252. Springer, Heidelberg (2008)
15. Microsoft. Live Mesh, https://www.mesh.com/welcome/default.aspx
16. Wang, L., Tao, J., Kunze, M.: Scientific Cloud Computing: Early Definition and Experience. In: Proceedings of the 2008 International Conference on High Performance Computing and Communications (HPCC 2008), pp. 825–830 (2008)