VCE - A Versatile Cloud Environment for Scientific Applications

Martin Koehler, Siegfried Benkner
Faculty of Computer Science, University of Vienna
Nordbergstr. 15/C/3, Vienna, Austria
koehler,sigi@par.univie.ac.at

Abstract—**Cloud computing promises to change the way scientists will tackle future research challenges by providing transparent access to distributed heterogeneous data sources and to high-end computing facilities for performing computationally demanding and data-intensive modeling, simulation and analysis tasks. In this article we describe the Vienna Cloud Environment (VCE), a service-oriented Cloud infrastructure based on standard virtualization and Web service technologies for the provisioning of scientific applications, data sources, and scientific workflows as virtual appliances that hide the details of the underlying software and hardware infrastructure. The VCE is based on a virtual appliance model within a component-based service provision framework, which supports the configuration of application, data, and workflow services from a set of basic service components providing capabilities like job or query execution, data transfers, QoS negotiation, data staging, and error recovery. The virtual appliance model constitutes an easy way to provision applications, data sources, and workflows in the Cloud and a standard way of accessing and integrating the appliances into client applications. VCE has been developed and utilized in the context of several projects for the realization of IT infrastructures within bio-medical and molecular modeling domains.**

***Keywords*-cloud infrastructure; software as a service; data mediation; workflow; MapReduce.**

I. Introduction

Cloud computing is often defined as realization of the ”Everything as a Service” model(XaaS) [1]. Cloud computing promises on demand access to virtually infinite compute and storage resources, programming and execution platforms, and applications, provided and consumed as services. More often the cloud computing stack is characterized as Software/Platform/Infrastructure as a Service (SaaS, PaaS, and IaaS). IaaS Clouds provide a shared infrastructure to their customers on a rental basis by utilizing virtualization technologies while programming and execution environments for Cloud programming models (e.g., Google AppEngine, MapReduce [2]) can be categorized as PaaS. On the top layer, Cloud computing enables transparent and dynamic hosting of applications together with their native execution environment without knowledge about the actual hardware infrastructure. If applications themselves provide a direct service interface to the user, this approach is called SaaS. The emergence of virtualization technologies and Cloud computing infrastructures (e.g., OpenNebula, Amazon EC2) enables easy provisioning of software as virtual appliances on remote resources on demand. A virtual appliance can be defined as a software package preinstalled on a virtual machine image to enable provisioning of the software in the Cloud. For utilizing Clouds in the scientific domain, additional capabilities are required for supporting long running applications, usually executed on HPC computing resources, and for extracting knowledge from huge data sets.

In this work we present the Vienna Cloud Environment (VCE), which follows the Software as a Service model to expose virtual appliances as services. VCE virtual appliances hide the details of the underlying software and hardware infrastructure and provide a common set of generic interfaces to the user. The service provisioning framework in VCE has been developed on top of the Vienna Grid Environment (VGE), a service oriented infrastructure for virtualizing scientific applications and data sources as Web services. VGE [3][4] was developed in context of the European projects GEMSS and @neurIST and has now been Cloud-enabled by supporting virtual appliances.

VCE enables the hosting of parallel high-performance computing applications, data sources, workflows, as well as data-intensive MapReduce applications in the Cloud by means of specific virtual appliances, all following the same generic service interface. Virtual application appliances support on demand access to high performance computing applications (e.g., parallel MPI/OpenMP codes) including support for dynamic negotiation of service-level-agreements based on a flexible QoS infrastructure using business models specialized for the application [5]. In addition to virtual application appliances VCE provides virtual data appliances to facilitate access to and integration of heterogeneous data sources. Virtual data appliances are built upon OGSA-DAI and OGSA-DQP and offer transparent access to multiple data sources via a virtual global schema relying on flexible data mediation techniques [6]. On top of application and data appliances, VCE offers support for scientific workflows [7]. Workflow appliances are based upon the WEEP Workflow Engine [8] and can be structured to adaptively load balance the workload across multiple appliances. In recent work support for MapReduce appliances was implemented within VCE [9]. MapReduce appliances include an adaptive framework for the execution of data-intensive Map-Reduce applications in the Cloud based on autonomic computing concepts.

The remainder of this paper is structured as follows. The next section describes the architecture of VCE. The following sections present in detail virtual appliances for applications, data sources, workflows, and MapReduce applications. The paper concludes with a summary and an outlook to future work.

II. VCE CLOUD INFRASTRUCTURE

The VCE Cloud infrastructure comprises a set of virtual appliances, a generic service provision framework, and a client-side Cloud programming environment. The service provisioning environment enables service providers to expose compute intensive scientific applications, distributed data sources, scientific workflows, as well as MapReduce applications as Cloud services that can be securely accessed on-demand by clients over the Internet. Virtual appliances include a preconfigured setup of the VCE service provisioning environment as well as appliance specific software (e.g., Workflow Enactment Engine). The client-side Cloud programming framework offers a high-level application programming interface (API) with Java, C# and C bindings that may be used to construct advanced applications from application, data, workflow, and MapReduce services.

A. VCE Architecture

VCE adopts a service-oriented architecture and relies on standard Web Service as well as Cloud computing technologies for offering parallel applications, distributed heterogeneous data sources, workflows, and MapReduce applications as virtual appliances. VCE distinguishes four different types of virtual appliances, application appliances, data appliances, workflow appliances, MapReduce appliances, which all are based on virtual images, Web service enabled via WSDL and securely accessed using SOAP messages.

Application appliances virtualize compute intensive applications, usually parallel MPI codes available on clusters, other HPC systems, or Cloud resources. Using application appliances, clients may run applications on demand over the Internet and negotiate with service providers required QoS levels, for example, to ensure that an application job is completed before a certain deadline. VCE application appliances provide generic Web service components for job execution, monitoring, data staging, error recovery and application-level quality of service support.

Data appliances virtualize data sources as Cloud services, facilitating transparent access to and integration of heterogeneous data sources including relational data bases, XML data bases and flat files. Relying on advanced mediation mechanisms, data appliances provide transparent access to distributed data sources via a single integrated virtual schema. VCE data appliances provide generic Web service components for query execution, data movement, and staging of result data.

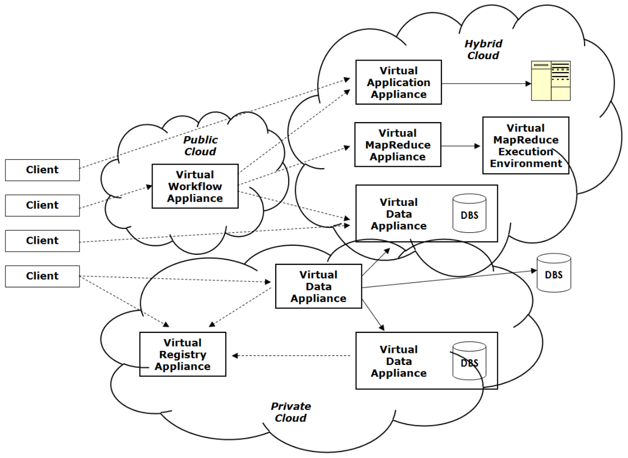
Hybrid Cloud
Virtual Application Appliance
Virtual MapReduce Appliance
Virtual MapReduce Execution Environment
Public Cloud
Virtual Workflow Appliance
Client
Client
Client
Client
Virtual Data Appliance
DBS
Virtual Data Appliance
DBS
Virtual Registry Appliance
Virtual Data Appliance
DBS
Private Cloud

Figure 1. A VCE Cloud comprising Application, Data, Workflow and MapReduce Services deployed in different Clouds

Workflow appliances virtualize scientific workflows as Cloud services, facilitating transparent access to and execution of scientific workflows. They are based upon the Workflow Enactment Engine (WEEP) and are able to schedule the workload to multiple workflow execution appliances. VCE workflow appliances provide generic Web service components for workflow execution, data movement, error recovery, and staging of result data.

MapReduce appliances virtualize data-intensive Hadoop applications, usually accessing huge data volumes stored in HDFS. MapReduce appliances internally use an adaptive execution framework based on autonomic computing concepts. The adaptive framework schedules the execution of the application automatically on available Cloud resources by considering the execution time as well as the number of resources. They provide generic Web service components for job execution, monitoring, data staging, and error recovery and are internally based upon the Hadoop framework.

As shown in Figure 1, a VCE Cloud usually comprises multiple application, data, workflow, and MapReduce appliances, as well as multiple client applications. Moreover, VCE offers an integrated certificate authority for providing an operational PKI infrastructure and end-to-end security based on X.509 certificates.

The VCE environment provides mechanisms for appliance discovery based on service registries. Multiple service registries may be set up in order to enable service providers to publish their services, and clients to discover these services. VCE service registries are realized as virtual appliances providing a Web service interface. Service providers can describe their services using an arbitrary set of attributes (name/value pairs), which are published in the registry. These attributes may be utilized during service selection to determine potential candidate services that might be able to fulfill a request.

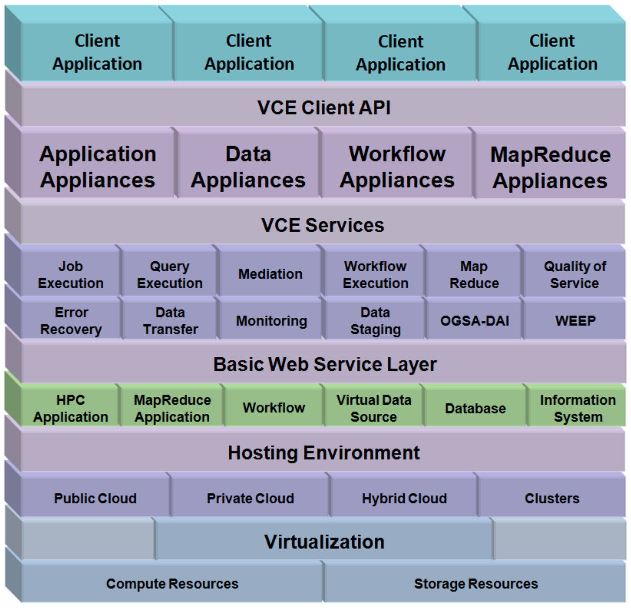

Figure 2. Layered abstraction of VCE

VCE virtual appliances, being virtual images, are hosted within Cloud environments. Different Cloud environments, as public Clouds (e.g., Amazon EC2) or private/hybrid Clouds can be utilized for the provisioning of virtual appliances. In several projects the following Cloud environments have been utilized: Eucalyptus, OpenNebula, and VMWare server. Virtual appliances provide access to one or more VCE services, which are hosted within a preconfigured service container. VCE client applications usually run on PCs or workstations connected to the Internet and make use of the VCE client API for interacting with services through the VCE middleware.

Figure 2 shows a layered abstraction of the VCE infrastructure. At the lower layer are a variety of compute and storage resources, which are Cloud-enabled via Cloud computing infrastructures or provided via schedulers as Sun Grid Engine (SGE). The HPC applications, scientific workflows, data sources and MapReduce applications are normally installed on VCE virtual appliances, but can be provided directly on the computing resources as well. These resources are virtualized through the Cloud service middleware layer and transparently provided through the abstraction of VCE services on virtual appliances, which are hosted on the Cloud resources. VCE services are composed of generic service components providing basic capabilities for job, query, and workflow execution, QoS, data mediation, data transfer, monitoring and error recovery. Client applications are able to transparently access VCE virtual appliances, through the abstraction of VCE services.

B. VCE Virtual Appliances

VCE virtual appliances provide a preconfigured installation of a service hosting environment based on the open-source frameworks Apache/Tomcat and Axis, a preconfigured VCE service, and a deployment tool for configuring and deploying VCE services. The VCE deployment tool automates the provision of HPC or MapReduce applications, data sources, and workflows as services. It offers an intuitive graphical or command line user interface for enabling service providers to describe, configure, deploy and manage services without having to deal with the details of Web service technologies.

The deployment tool enables to specify the service hosting environment, the security level, and a service description. The description of a service usually comprises the specification of input/output file names and of scripts for starting job execution, or the execution of a scientific workflow. A description of a data service comprises the specification and configuration of the underlying data sources. VCE services support different security levels, including no security, basic security via SSL, and end-to-end security. Supported security technologies include PKI, HTTPS, WS Security and an end-to-end security protocol for separate encryption of sensitive portions of the transferred data. The information specified by the user with the deployment tool is stored internally in an XML service descriptor. Upon deployment of a service, a Web service with a corresponding WSDL interface is generated, deployed in the VCE hosting environment, and published in the VCE registry.

A virtual appliance additionally includes an Apache server for connecting VCE services with the Internet using a Tomcat connector (JK connector) between the Apache server and the Tomcat server. Additionally the appliance includes a preconfigured firewall enabling access via ssh and http.

C. Virtual Appliance Access Model

VCE relies on a purely client-driven approach for accessing VCE virtual appliances via SOAP. All interactions of a client with appliances are initiated by the client and neither call-backs nor notification mechanisms are used.

VCE appliances are inherently multi-threaded, i.e., if multiple clients access a service, a separate thread is generated for each client, and for each client a separate application/workflow job or data access is generated. Session management and state handling is managed internally and transparently, conceptually following the WSRF model but being implemented based on conversational identifiers and WS-Addressing mechanisms.

A basic VCE appliance access scenario starts with administrative steps including authorization and authentication. The client then usually accesses a registry to find a set of candidate appliances. Afterwards the client uploads the input data, initiates execution, queries for the state of the execution, and finally downloads the result.

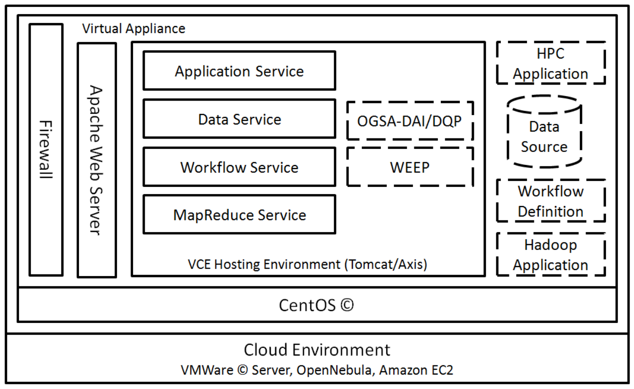

Figure 3. Architecture of Virtual Appliance

D. VCE Client Environment

In order to support the construction of client-side applications that interact with virtual appliances, VCE provides a high-level client API with bindings for JAVA, C and C# (.NET). The client API hides most of the details of dealing with remote services from the client application developer. The central abstraction provided by the client API is a ServiceProxy interface, which specifies all methods required for transparently accessing application and data services from a client application. Moreover, the client API provides a set of classes for dealing with various aspects of the VCE environment at a high-level of abstraction hiding the details of the underlying interaction with VCE.

The client API is structured into several layers. The bottom layers implement data marshaling, message generation, signing, and encryption, while the top layers provide abstractions for service discovery and all service interactions provided by the ServiceProxy. Using these high-level abstractions, users can more easily develop client applications that interact with VCE appliances. The ServiceProxy interface comprises methods for handling transfers of input and output data, for job, workflow or query execution, for monitoring, as well as for error handling.

Additionally, VCE clients support several security mechanisms as provided by VCE and monitoring of the execution. The high-level client API is based on the service component model and therefore extensible with new service component functionalities.

III. Virtual Application Appliances

The VCE enables service providers to virtualize HPC applications available on clusters, other parallel hardware, or virtual images as application appliances that can be accessed on-demand by clients over the Internet. Application appliances hide the details of their execution environment, providing abstract interfaces for managing job execution on remote computing resources. The VCE application appliance infrastructure has been partially developed within the EU Project GEMSS [10], which devised a service-oriented Grid infrastructure that supports the Grid provision of advanced medical simulation. In order to enable the utilization of Grid-based simulation services during medical procedures, QoS support to ensure the timeliness of simulation results was a major requirement. Addressing these issues, VCE application appliances may be configured with a flexible QoS negotiation service component, supporting dynamic negotiation of Service Level Agreements (SLAs) [5]. The QoS infrastructure enables clients to negotiate guarantees on service response times and price with service providers.

Virtual application appliances can be provided by directly installing the application on the appliance. In this case an application can be distributed to service providers by virtual application appliances and exposed as Web services by hosting the appliance in the Cloud. Virtual application appliances are able to utilize a scheduling system for the execution of jobs. Therefore it is possible to access applications installed on local cluster resources via Sun Grid Engine (SGE) as well as applications deployed on Cloud resources via VCE application appliances.

IV. Virtual Data Appliances

Virtual data appliances were partly developed in the context of the European project @neurIST [11], which developed an advanced service-oriented IT infrastructure [12][6] for the management of all processes linked to research, diagnosis and treatment development for complex and multifactorial diseases.

To support such scenarios, VCE includes a generic data management and integration framework for the provision and deployment of virtual data appliances. VCE enables the virtualization of heterogeneous scientific databases and information sources as Web services hosted in virtual appliances, which allows transparent access to and integration of relational databases, XML databases and flat files. The development of virtual data appliances utilizes advanced data mediation and distributed query processing techniques based on GDMS [13], OGSA-DAI [14], and OGSA-DQP [15].

The VCE offers virtual data appliances as well as virtual mediation appliances, all providing the same interface to clients. Virtual data appliances (VDA) provide access to a single data source, and virtual mediation appliances (VMA) offer transparent access to multiple data sources via a global virtual schema. The virtual schema of a VMA provides an integrated, global view of the underlying local data sources. Virtual mediation appliances translate queries with respect to the global schema into local queries, manage the access to the local data sources, and integrate results from local queries according to the global schema. Different, tailor-made views of distributed data sources may be provided for specific usage-scenarios, by setting up different variants of virtual mediation appliances.

Virtual mediation appliances can be composed recursively, i.e., a VMA can act as a local data source to another virtual mediation appliance, resulting in a tree-structured data integration scenario. In order to optimize complex data integration scenarios, virtual mediation appliances may be configured to support distributed query processing. For distributed query processing, the data mediation engine relies on OGSA-DQP, which has been integrated to coordinate the distributed execution of queries to local data sources utilizing a set of additional virtual evaluation nodes, as specified by the service provider.

Virtual data and mediation appliances are based on a virtual images and encapsulate a VCE service provisioning installation providing access to the underlying data source. A fully configured system installation is provided as a virtual machine, including preconfigured system components such as Apache, Tomcat and SSL configuration. This allows simple deployment for test and production use in Cloud computing infrastructures. Virtual data appliances includes a preconfigured data access service providing a Web service interface to a preinstalled MySQL database. Virtual mediation appliances include a preconfigured installation of a data mediation service with OGSA-DQP support. The service provider has to provide the mediation schema to the virtual appliance to configure the service. Both virtual appliances are available for VMWare and Eucalyptus and are based on CentOS.

V. Virtual Workflow Appliances

Support for virtual workflow appliances has been implemented in the context of the CPAMMS project. A major challenge in this project is the provisioning of seamless integrated support for scientific workflows in the domain of computational molecular modeling, which access HPC applications and are deployed on globally distributed computing resources. These scientific workflows are typically long running, deal with huge files, and have a need for dynamic execution control mechanisms. We offer support for scientific workflows in the VCE by seamlessly integrating the Ubuntu Cloud infrastructure supporting the scheduling of dynamic and partitioned workflows deployed on virtual workflow appliances. In [16], [7] we described virtual workflow appliances in detail and a case study workflow for computing photodynamics of biologically relevant molecules.

Due to the dynamic characteristics and because of the unpredictable runtime and resource requirements of this workflow, we implemented a framework for the provisioning of scientific workflows as virtual workflow appliances in the Cloud, where resources can be made available on demand. We integrated a workflow enactment engine into the VCE and prepared the middleware for hosting workflow services transparently in the Cloud. Following the Cloud and Web service paradigms, a virtual workflow appliance for hosting scientific workflows exposed as Web service has been developed. The virtual workflow appliance includes all required software packages allowing an easy deployment of new scientific workflows as Web services by leveraging Cloud technologies. Scientific workflows usually include many potentially long running scientific codes, each of them exposed as a VCE service, which may be invoked many times during a specific workflow. To make use of the dynamic resource allocation possibilities in the Cloud, our virtual workflow appliances are able to delegate these requests to basic service invocation appliances. This leads to a decentralized execution of the workflow in the cloud environment and allows to schedule the different service invocations to different instances of the appliance.

The basic service invocation workflow encapsulates the life cycle of accessing a VCE appliance and is deployed as WEEP workflow [8] in the basic service invocation appliance. A simple interface is provided by the service invocation workflow, which can be used by workflow designers during the workflow development process. The basic workflow constitutes an additional abstraction layer hiding the details of accessing and querying a specific VCE appliance from the workflow designer. Multiple instances of the basic service invocation appliance are hosted in the Cloud environment and can be used to realize a distributed execution of different service invocations.

The virtual workflow appliance can be used to deploy scientific workflows as VCE services without the need of additional software installation. The virtual workflow appliance is configured with a VCE service provisioning environment. The service provisioning environment includes a preconfigured VCE workflow service definition, which allows the deployment of a workflow service simply by the provisioning of a WEEP Grid workflow archive (gwa) including the BPEL process definition. Using the VCE deployment tool, the VCE workflow service can be deployed automatically by providing the reference to the used gwa. The deployed VCE service comprises a workflow service component, virtualizing a workflow behind the generic VCE interface, and a streaming service enabling direct file transfers between services based on a push/pull mechanism. The workflow service automatically deploys the provided gwa in a local WEEP Engine installation accessible via a JMX interface. A WEEP template service is instantiated for the gwa, and a engine service is created, when the workflow is started. It is possible to host several VCE workflow services in one virtual workflow appliance. The VCE workflow appliance is configured with an advanced WebPortal allowing live logging, online user management, and a push/pull mechanism supporting large data transfers directly between services.

VI. Virtual MapReduce Appliances

Virtual MapReduce appliances [9] enhance the VCE with support for data-intensive applications based on the Apache

Hadoop framework. Virtual MapReduce appliances include an installation of the VCE service provisioning environment and expose a VCE MapReduce service via a Web service interface. VCE MapReduce services use an integrated self-configuring and adaptive framework for optimizing the configuration of data-intensive applications at three abstraction layers: the application layer, the MapReduce layer, and the resource layer. For each layer generic descriptors, describing the evolving parameter set, are introduced. At the application layer the framework is able to configure the execution of parameter studies, which results in different resource needs of the application. The Hadoop framework provides a huge set of configuration parameters and the installation has to be optimized for the actual set of resources [17]. Using a MapReduce descriptor, the framework is able to automatically determine an optimized setting of Hadoop parameters, as for example the number of map and reduce tasks, on a per job basis. Leveraging a scheduler, in our implementation the Sun Grid Engine (SGE), allows describing and selecting the available computing resources by taking into account the actual resource needs for a job.

The adaptive framework evaluates the arising parameter set and self-configures all layers on a per job basis. It is based on the MAPE-K loop [18][19], which is a well-known concept from autonomic computing, and a utility function [20]. In recent work [9], we evaluate the possible configurations based on a history based performance model and a utility function. By adapting the utility function, the system allows to specify the main optimization goal, optimizing resource usage or optimizing the runtime of a job.

Virtual MapReduce appliances rely internally on a scheduling system for the execution of MapReduce applications on remote resources. In [9], we leverage the OGE to schedule applications on local cluster systems, but we additionally provide MapReduce appliances to enable dynamic hosting in Cloud environments.

VII. Related Work

Over the last years, the scientific community has undertaken a lot of effort for enabling the utilization of Clouds for science and for the provisioning of scientific Clouds. Some institutions provide access to their scientific Clouds at `ScienceClouds.org` [21]. They provide compute and storage capability to all scientists wanting to run their applications in the Cloud. On the other hand, there is a lot of effort in the creation of open source Cloud computing toolkits. These Cloud computing toolkits can be utilized by institutions for setting up private, public, or hybrid Clouds which can be utilized by domain scientists. The OpenNebula project [22] developed a software stack for the provisioning of Infrastructure as a Service (IaaS) Clouds. The VCE, can be utilized on top of IaaS Clouds, and provides an easy and generic way for Cloud-enabling scientific applications, data sources, and workflows as Cloud services following the SaaS concept. In [23], a discussion of benefits and issues regarding to science Clouds is provided. Identified as main benefits are the virtual ownership of resources as well as the ease of deployment, while some open issues include performance management, interoperability, and execution models.

The industry provides different Cloud computing solutions, most often on a pay per use basis. The Amazon EC2 Cloud [24] provides IaaS services to their customers based on different instance types. They support specialized services for HPC computing including Amazon Elastic MapReduce and Public Data Sets. Microsoft provides with Windows Azure [25] services for hosting and scaling of Web applications on their data centers. Google follows the PaaS approach by providing a high-level and scalable programming API to their users (Google App Engine [26]). Many more commercial service providers are supporting different aspects of the SaaS/PaaS/IaaS stack.

VIII. Conclusion and Future Work

In this paper we presented the Vienna Cloud Environment, a generic Cloud infrastructure for virtualizing HPC applications, data sources, scientific workflows, and MapReduce applications as virtual appliances in the Cloud. Virtual appliances hide the details of the underlying software and hardware infrastructure and can be easily distributed and deployed. VCE application appliances virtualize HPC applications running on clusters by providing common operations for transparently managing the execution of application jobs. Virtual data and mediation appliances address the complex problems associated with access to and integration of distributed heterogeneous data sources. Virtual workflow appliances add support for distributed execution of complex long running scientific workflows. They include a preinstalled workflow engine and allow the deployment and execution of composed workflows written in BPEL. A basic adaptive execution framework for MapReduce applications has been realized for virtual MapReduce appliances. By utilizing virtual appliances, scientists are enabled to run their applications on virtually infinite resources in the Cloud and to easily access them via a Web service interface.

Our future work will focus on improved adaptive execution mechanisms based on autonomic computing concepts addressing the trade-offs between execution time, resource requirements/costs for Cloud-based scientific applications.

References

[1] "Everything as a Service," March 2011, http://www.hp.com/hpinfo/execteam/articles/robison/08eaas.html.

[2] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, pp. 107–113, January 2008. [Online]. Available: http://doi.acm.org/10.1145/1327452.1327492

[3] S. Benkner, I. Brandic, G. Engelbrecht, and R. Schmidt, "VGE - A Service-Oriented Grid Environment for On-Demand Supercomputing," in *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (Grid 2004)*, November 2004.

[4] S. Benkner, G. Engelbrecht, M. Koehler, and A. Woehrer, "Virtualizing scientific applications and data sources as grid services," in *Cyberinfrastructure Technologies and Applications*, J. Cao, Ed. Nova Science Publishers, 2009.

[5] G. Engelbrecht and S. Benkner, "A service-oriented Grid environment with on-demand QoS support," in *ICWS-2009 - 7th IEEE Int. Conference on Web Services*, Los Angeles, CA, USA, 7 2009.

[6] M. Koehler and S. Benkner, "A service oriented approach for distributed data mediation on the grid," in *Grid and Cooperative Computing, 2009. GCC '09. Eighth International Conference on*, Lanzhou, Gansu, China, August 2009, pp. 401–408. [Online]. Available: http://dx.doi.org/10.1109/GCC.2009.35

[7] M. Koehler, M. Ruckenbauer, I. Janciak, S. Benkner, H. Lischka, and W. N. Gansterer, "A grid services cloud for molecular modelling workflows," *International Journal of Web and Grid Services (IJWGS)*, vol. 6, no. 2, pp. 176 – 195, 2010.

[8] I. Janciak, C. Kloner, and P. Brezany, "Workflow Enactment Engine for WSRF-Compliant Services Orchestration," in *In The 9th IEEE/ACM International Conference on Grid Computing*, 2008.

[9] M. Koehler, Y. Kaniovskyi, and S. Benkner, "An adaptive framework for the execution of data-intensive MapReduce applications in the Cloud," in *DataCloud2011 - 1st Int. Workshop on Data Intensive Computing in the Clouds*, Anchorage, Alaska, USA, 5 2011.

[10] S. Benkner, G. Berti, G. Engelbrecht, J. Fingberg, G. Kohring, S. Middleton, , and R. Schmidt, "Gemss: Grid-infrastructure for medical service provision," in *Proceedings of HealthGRID 2004*, January 2004.

[11] H. Rajasekaran, P. Hasselmeyer, L. L. Iacono, J. Fingberg, P. Summers, S. Benkner, G. Engelbrecht, A. Arbona, A. Chiarini, C. Friedrich, M. Hofmann-Apitius, B. Moore, P. Bijlenga, J. Iavindrasana, H. Müller, R. Hose, R. Dunlop, A. Frangi, and K. Kumpf, "@neurIST - Towards a System Architecture for Advanced Disease Managment through Integration of Heterogeneous Data, Computing, and Complex Processing Services," in *IEEE International Symposium on Computer-Based Medical Systems*. Jyväskylä, Finland: IEEE Computer Society Press, June 2008, copyright (C) IEEE Computer Society.

[12] S. Benkner, A. Arbona, G. Berti, A. Chiarini, R. Dunlop, G. Engelbrecht, A. F. Frangi, C. M. Friedrich, S. Hanser, P. Hasselmeyer, R. D. Hose, J. Iavindrasana, M. Köhler, L. L. Iacono, G. Lonsdale, R. Meyer, B. Moore, H. Rajasekaran, P. E. Summers, A. Wöhrer, and S. Wood, "@neurist: Infrastructure for advanced disease management through integration of heterogeneous data, computing, and complex processing services," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 14, no. 6, pp. 1365–1377, November 2010.

[13] A. Wöhrer, P. Brezany, and A. M. Tjoa, "Novel mediator architectures for grid information systems," *Future Generation Computer Systems*, vol. 21, no. 1, pp. 107 – 114, 2005.

[14] M. Antonioletti, M. Atkinson, R. Baxter, A. Borley, C. Hong, P. Neil, B. Collins, N. Hardman, A. C. Hume, A. Knox, M. Jackson, A. Krause, S. Laws, J. Magowan, N. W. Paton, D. Pearson, T. Sugden, P. Watson, and M. Westhead, "The design and implementation of grid database services in ogsa-dai: Research articles," *Concurrency and Computation : Practice and Experience*, vol. 17, no. 2-4, pp. 357–376, 2005.

[15] M. N. Alpdemir, A. Mukherjee, A. Gounaris, N. W. Paton, P. Watson, A. A. Fernandes, and D. J. Fitzgerald, "Ogsa-dqp: A service for distributed querying on the grid," in *Advances in Database Technology - EDBT 2004*, ser. Lecture Notes in Computer Science, E. Bertino, S. Christodoulakis, D. Plexousakis, V. Christophides, M. Koubarakis, K. Böhm, and E. Ferrari, Eds. Springer Berlin / Heidelberg, 2004, vol. 2992, pp. 3923–3923, 10.1007/978-3-540-24741-8_58.

[16] M. Ruckenbauer, I. Brandic, S. Benkner, W. Gansterer, O. Gervasi, M. Barbatti, and H. Lischka, "Nonadiabatic Ab Initio Surface-Hopping Dynamics Calculation in a Grid Environment - First Experiences," in *Proceeedings of the 2007 International Conference on Computational Science and Its Applications (ICCSA 2007)*, ser. LNCS, vol. 4705. Kuala Lumpur, Malaysia: Springer Verlag, 2007, pp. 281–294.

[17] Impetus, "Whitepaper: Deriving intelligence from large data using hadoop and applying analytics," March 2011.

[18] J. Kephart and D. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41 – 50, Jan. 2003.

[19] M. C. Huebscher and J. A. McCann, "A survey of autonomic computing - degrees, models, and applications," *ACM Comput. Surv.*, vol. 40, pp. 7:1–7:28, August 2008. [Online]. Available: http://doi.acm.org/10.1145/1380584.1380585

[20] W. Walsh, G. Tesauro, J. Kephart, and R. Das, "Utility functions in autonomic systems," in *Autonomic Computing, 2004. Proceedings. International Conference on*, May 2004, pp. 70 – 77.

[21] "ScienceCloud," March 2011, http://scienceclouds.org.

[22] "OpenNebula," March 2011, http://opennebula.org.

[23] C. A. Lee, "A perspective on scientific cloud computing," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ser. HPDC '10. New York, NY, USA: ACM, 2010, pp. 451–459. [Online]. Available: http://doi.acm.org/10.1145/1851476.1851542

[24] "Amazon elastic compute cloud (amazon ec2)," March 2011, http://aws.amazon.com/ec2.

[25] "Windows Azure - Microsoft's Cloud Services Platform," March 2011, http://www.microsoft.com/windowsazure/.

[26] "Google AppEngine," March 2011, http://code.google.com/appengine/.