Building an On-Demand Virtual Computing Market in Non-Commercial Communities

Matthias Steinbauer
Department of Telecooperation, Johannes Kepler University Linz
Altenbergerstraße 69
Linz, Austria
matthias.steinbauer@jku.at

Ismail Khalil
Department of Telecooperation, Johannes Kepler University Linz
Altenbergerstraße 69
Linz, Austria
ismail.khalil@jku.at

Gabriele Kotsis
Department of Telecooperation, Johannes Kepler University Linz
Altenbergerstraße 69
Linz, Austria
gabriele.kotsis@jku.at

ABSTRACT

This work describes a system that enables non-commercial communities, e.g. students and researchers, to deploy preconfigured clusters of virtual machines on arbitrary cloud computing stacks implementing the Open Cloud Computing Interface (OCCI). As a key enabling technology, the virtualization and provisioning available in such infrastructure as a service clouds is leveraged in order to instantiate and setup clusters on a user's demand. A simple, web-based market system is provided as the user interface. It allows the browsing, configuring and launching of different clusters of appliances. Running virtual machines are accessible directly from the market via integrated remote desktop software.

Categories and Subject Descriptors

D.1.3 [**Programming Techniques**]: Concurrent Programming—*distributed programming*; D.4.7 [**Operating Systems**]: Organization and Design—*interactive systems, distributed systems*

General Terms

Design, Experimentation, Management

Keywords

Cloud Computing, Cluster, Hadoop, Market, Provisioning, Storm

1. INTRODUCTION

Cloud computing as a discipline of computer science [14] is being adopted widely and in many different application areas. Technologies are considered mature for practical applications, thus, leaving room for application centric research in this area and creating a demand for educational courses on cloud computing.

SAC'13 March 18-22, 2013, Coimbra, Portugal.

Applications are based on different technologies. Starting at providing complete virtual computing instances, that may become provisioned via software interfaces, called Infrastructure as a Service (IaaS). Other systems provide complete computing Platforms as a Service (PaaS), often using IaaS as its base layer in order to build large clusters supplying applications with computing needs that are not available in a single computing node. Especially, Big Data systems are often implemented as PaaS. On a further service level, single applications may be made available as a service, called Software as a Service (SaaS).

For all of the presented service levels, there are many different implementations available. They often share similar concepts, but use different implementations. In order to study different aspects of cloud computing systems, obviously, one needs to have access to them.

We observe that many labs already have access to their own cluster systems as test-beds or there is work in progress to install one shortly. This is because specific tasks, like testing different PaaS frameworks or running many virtual machine images for educational purposes, need specific testbeds. We also observe that often the setup of such testbeds is very time consuming and costly. Not only in terms of hardware and setup of infrastructure clouds, but also in terms of setting up virtual appliances and PaaS clusters.

Since usually only one group is able to access its own computing resources, this locks the variety of research experiments down to the limits of the locally installed cloud infrastructure. In a typical research and educational institution, there is a further trade-off in sharing resources for research and educational purposes.

The objective of this paper is to propose a solution to the described issues in the following ways: (1) The sharing of computing resources in a non-commercial oriented way, (2) providing easy access and configuration to these computing resources and (3) enabling one-click deployment of different SaaS and PaaS systems upon IaaS clouds.

Prospective benefits from such a system would be that groups are then able to scale out to off-promise clouds if the on-promise computing resources are too limited. Setup times for PaaS and SaaS system could be reduced since preconfigured appliances and clusters eliminate the need to configure each single instance of such a system. Members of different institutions would be able to share appliances and their configuration with others in the market.

During our research we focus on two specific examples

that we use in a classroom scenario. In order to support research and lectures covering the Hadoop [2] and Storm [5] platforms we need test-beds for those two systems. Ideally the test beds for different use cases are separated entirely. This means that for each project in classroom or research a new setup of the used platform is required. The configuration and size of the clusters used in different setups vary very much. Starting from single node development environments, which are supporting use cases in early project phases. Up to full blown cluster setups that are using many compute nodes in order to distribute computational workload or data. In order to illustrate our approach we use examples from our Hadoop and Storm clusters throughout this paper.

In section 2, one may find a comparison of related work. Section 3 describes a generic concept for supplying non-commercial communities with virtual computing resources. In section 4, we will explain our prototype implementation. In section 5, starting points for future work are elaborated. Finally, in section 6, we conclude on our work.

2. RELATED WORK

Allowing easy access to computing resources is a common goal for many cloud computing sites and projects. One example that focuses on creating a test-bed for research is OpenCirrus [6]. It is a middle-ware that is capable of providing access to four different cloud computing sites and their resources. The available resources are virtual machines and bare metal in heterogeneous clusters. This system provides computing resources, but does not provide preconfigured virtual machine images or clusters.

In contrast, the myHadoop project focuses on the provisioning of Hadoop clusters. It provides tools and mechanisms in order to deploy Hadoop clusters on traditional high performance computing sites. This way the system leverages tools from high performance computing in order to deploy the Hadoop cluster. In [11] it is also briefly discussed that mileage may vary since in high performance computing, compute nodes with minimal local storage are usually used whereas Hadoop or other programming models, similar to MapReduce, are designed for shared-nothing architectures.

The goal of Future Grid and its educational virtual clusters [8] is to provide easy to use virtual appliances that can be configured as clusters. The focus of this system is entirely on training. The users should have very short turn around times on setting up their system. This is achieved by providing a very basic Linux based appliance that can be provisioned on demand. Once the user has access to the virtual machine, he is able to install and start more complex cluster systems by executing batch scripts inside the appliance. This way, single cluster nodes are configured automatically.

Of course, the process could be further simplified by also automating the provisioning of complete clusters, i.e. not only automating the setup and basic configuration of software, but automating the whole process. This would be achieved by creating the virtual instances, installing the necessary software and, finally, configuring each node according to its task. Thus, enabling one-click provisioning of complete cluster systems. As explained in [9], this process is often automated, however, mostly only the provisioning of one type of cluster system is supported. For instance, there exist may different ways for one-click provisioning of Hadoop systems [1, 3, 4]. Our system, however, is able to harness several different cluster systems and provides access to them with one generic user interface.

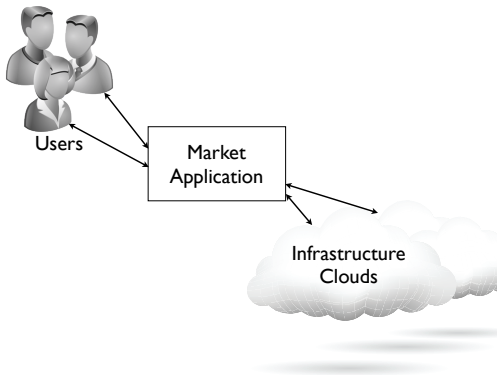

Figure 1: Overall architecture of a system that is providing on demand computing resources to a scientific user. The user is using a SaaS application that functions as a middle-ware in order to serve his computing needs. The middle-ware is capable of acquiring computing resources from different IaaS providers.

3. CONCEPT

We propose a market application implemented as SaaS that allows users to configure and access various infrastructure clouds in an easy way.

In figure 1, one may view the proposed, overall system design. The user is interacting directly with an SaaS application that we will later call, the market. This application holds virtual machine images, host and network configurations. The configuration interface is also capable of adapting the configuration of machines during start-up time and, thus, not only configuring single virtual machines. The middle-ware is able to configure complete clusters for more complex PaaS or SaaS systems that consist of many computing nodes.

Computing resources that were instantiated as a cluster resource are also configured for further provisioning. Meaning, the user is able to add or remove more computing resources to his clusters as needed.

In the development of different cloud computing stacks, the demand for common protocols to access them also arises. Some protocols receive a de facto standard since they are widely used in the industry. This would be, for instance, Amazons EC2 cloud API, which has also been adopted by other systems like OpenNebula or Eucalyptus [12, 15]. In this way, applications are able to run on private clouds, but could be scaled out to public cloud service providers.

The Open Grid Forum, on the other hand, is publishing a specification of the Open Cloud Computing Interface (OCCI) [13]. It is a protocol set that enables IaaS cloud providers to open their system to consumers. OCCI is designed to provide interoperability between cloud based applications and different IaaS providers, enabling the creator of applications to scale out on different clouds from different vendors. To gain useful benefit from such type of cloud middle-ware, it will not be sufficient to implement only the OCCI standard. Several cloud stacks and vendors often use

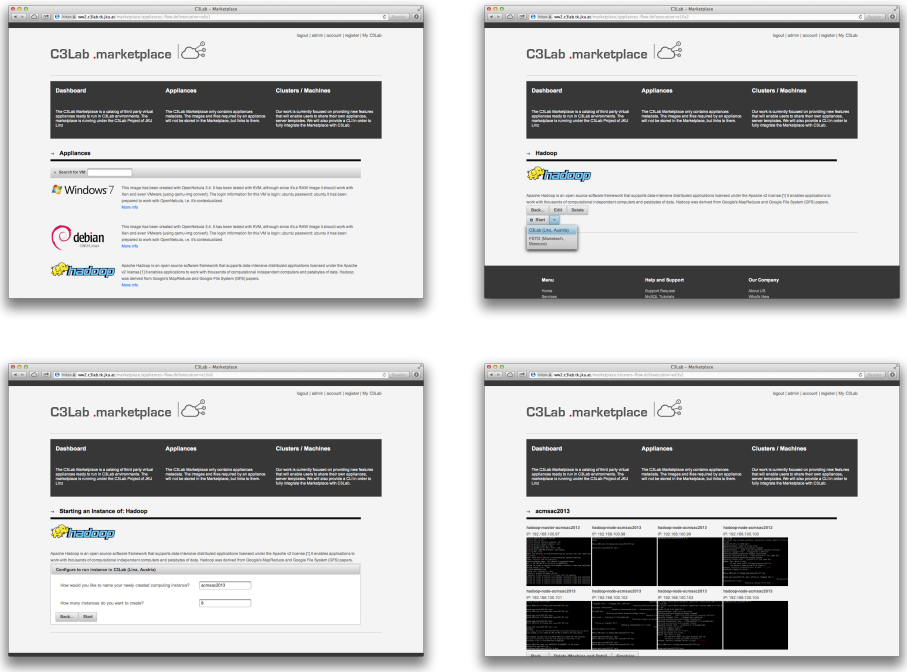

Figure 2: Screenshots of a users work-flow: After (1) selecting the Hadoop appliance and (2) deciding for a data center at which to instantiate and run the cluster, one is presented with (3) a simple configuration dialog.For clustered appliances, one may choose the number of virtual instances which are to start. After a cluster has started up, users can (4) directly access each single node directly from the application.

Amazons EC2 API which is evolving to a industry standard. Creating the requirement that the middle-ware layer is capable of talking different cloud access protocols in order to gain wider support of cloud stacks.

A system that provides on-demand computing resources to non-commercial communities needs to be able to use these different protocols in order to create and instantiate computing resources as requested by the user. The application that provides access to the computing resources itself has to be implemented as a SaaS application allowing web based access.

This application functions as a middle-ware and as a mediator in several ways. First, it provides a common interface to preconfigured computing resources that are easily deployed in different data centers. The user does not need to know the specifics of each data center, such as the installed IaaS system.

The application is also hosting virtual machine images, network and host configurations so the user does not need to know the specifics of configuring the PaaS or the SaaS system he is planning to use.

Finally, from a service contracts point-of-view, the user does not need to have individual contracts with several cloud service providers in order to use different clouds in different data centers - the provider of the middle-ware is responsible for that task.

4. IMPLEMENTATION

In order to prove the concept explained in section 3, we have created a prototype that addresses the issues, which were discussed. This prototype is implemented as a Java Enterprise application that may be deployed in an arbitrary Java Servlet Container.

This application provides a market to interested users by hosting different virtual machines images and their configurations. It is currently connected to two IaaS systems running the OpenNebula software stack with the OCCI API enabled. The system is enabled to connect to many different cloud infrastructures and could be easily adapted to also support other protocols besides OCCI.

The market is assuming the use of preconfigured virtual machine images and, thus, does not provide extensive configuration mechanisms to create virtual machine images from scratch. The virtual machine images are currently pre installed on the respective infrastructures. In order to test run our system, we currently provide a virtual machine image with Debian 6.0 and one with Windows 7, both of which are configured automatically and accessible directly from the market application [1] via the VNC or RDP protocols.

For more complex cluster systems, it is also possible to start many machines at once. We currently use a OpenNebulas contextualization [10] feature in order to configure the virtual machines and virtual clusters during boot time.

In order to test-run the cluster provisioning feature, we created two preconfigured images that contain all the software necessary to create and run Hadoop [2] and Storm [5] clusters. The market application is able to fully configure

[1]The prototype application and its source code along with further instructions are ready for download at `http://www.steinbauer.org/publish/vm-market.zip`

and start those images on arbitrary cloud computing infrastructures.

In figure 2, one may view screenshots of our market application. A user selected and configured an eight-node Hadoop cluster and monitors its consoles directly from the web-based application.

4.1 Hadoop

Hadoop is a PaaS level cloud computing platform designed for Big Data processing. It is an open source implementation of Google's distributed file-system and MapReduce programming model [7]. This allows for the creation of very data centric applications that are using the distributed file system as their storage. MapReduce applications usually run as batch processing jobs, reading and writing to the distributed file system.

The system is very elastic or scaleable and consists of several components, which may run on different machines. Particularly important is the NameNode, which provides the index of the distributed file-system. Usually, one also runs a MapReduce TaskTracker on the same node. The TaskTracker is able to schedule MapReduce jobs and sends them to worker nodes, called JobTracker, in Hadoop. Nodes running the JobTracker usually also run the DataNode component that is used to manage chunks of the distributed file system.

With our images, the user is able to run the following two configurations: (1) a development setup and a (2) a production setup.

During development, one could run all four components on one single machine. One would do that during development to debug MapReduce jobs or to investigate the Hadoop distributed file system. A user is able to achieve this setup by simply starting a one-node Hadoop cluster in our market.

During production use of Hadoop, the system is usually deployed in a cluster setup. The most common setup is to have one dedicated machine that is running the NameNode and TaskTracker components of Hadoop. This is the master node of the cluster and a single point of failure for the system. This master node is accompanied with many slave nodes, that run the DataNode and JobTracker components. Thus, every slave node is able to store parts of the distributed file system and execute MapReduce jobs on its local data. In this setup, replication is also enabled such that slave nodes may fail without influencing the stability of the complete system. We are able to create that type of clusters on demand within our market by instantiating clusters of any size. If more than one node is assigned to a Hadoop cluster, the first node is a designated NameNode and TaskTracker - all other nodes become configured as DataNodes and JobTrackers.

In very large setups, having the NameNode and the TaskTracker component on the same host may be a performance issue. To overcome this, it would also be possible to run the NameNode and the TaskTracker on different machines, however, this is not supported for automatic provisioning by our system right now. Nonetheless, the user has access to each individual computing node and, thus, is able to reconfigure his cluster manually for such a setup.

4.2 Storm

As a second test case, we use Twitter Storm, which is also a PaaS cloud computing platform. In contrast to Hadoop, this Big Data system is focused on streaming data. It allows the creation of topologies of computing elements, which get distributed on a Strom cluster. The cluster management system takes care of load balancing and routing data correctly between the individual processing elements.

Since Storm clusters are designed to not save a persistent state to a file system, the system does not need any tools to govern a distributed file system. This leads to a system where only two types of nodes exist: The Nimbus node, which is the master node of a Storm cluster and an arbitrary number of worker nodes, which are running a process called Supervisor.

4.3 Virtual machine contextualization

In order to configure single virtual machines or multi-node clusters, OpenNebulas' contextualization feature is leveraged. This is done by creating a virtual CD-Rom image on demand by the infrastructure. This image contains a configuration file and user scripts as specified in the virtual machine template. The configuration file is also created from the infrastructure services and contains, for instance, the IP address of the currently created virtual machine or its host-name. It is also possible to add custom configuration variables in order to perform more complex configurations. Similar configuration mechanisms are available within other infrastructure clouds.

For basic Debian 6.0 and Windows 7 virtual machines, only the networking and users inside the virtual machine are configured as specified during provisioning in the market application. For the more complex cluster setup of Hadoop or Strom clusters, further contextualization parameters are used to completely configure the cluster system.

We introduced a context parameter called *MASTER*. If this variable is set for a virtual machine instance, it signals to the configuration scripts that the current instance is to be configured as a master node. If the variable is not available and the virtual machine image is a cluster image, then the node is configured as a worker node. In the worker node setup, the configuration scripts look for a parameter called *IP_MASTER*, specifying the master node they should be assigned to.

The virtual machine images for Hadoop and Storm clusters are preloaded only with the needed software and are the same for each type of node. Whether a virtual machine is to become a master or worker node is decided during boot time [2].

The market application is using a two phase instantiation strategy, depicted in figure 3, in order to configure clusters and hosts correctly. In a first instantiation run, a virtual machine template is instantiated and the market specifies the *MASTER* parameter for the virtual machine context. Next, the template is submitted to the OCCI service and the market waits for the underlying cloud infrastructure to initialize the virtual machine. After this is done, the IaaS layer has assigned an IP address to the newly created virtual machine. This IP address is then queried via web service and used to create the templates for the worker nodes. The IP address of the master host is specified in the *IP_MASTER* parameter. Those templates are then submitted to the OCCI service in a batch.

[2]Please find the virtual machine images and the configuration scripts ready for download at `http://www.steinbauer.org/publish/vm-context-hadoop-storm.zip`

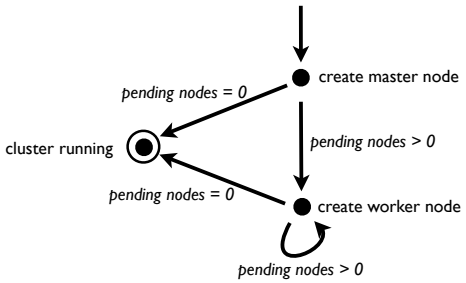

Figure 3: The first virtual machine is configured as the master node separately from the other machines in a cluster. If the master machine was created correctly, all other machines are configured in a batch.

4.4 Virtual machine setup

Besides the pure setup of the software inside a virtual machine, its machine image also needs to be prepared for contextualization. As explained earlier, OpenNebula uses virtual CD drives in order to map information from the cloud infrastructure into the virtual machine during boot time. We have created several scripts to make our cluster templates work with that technology.

First, the virtual image is modified in such a way that during boot time the virtual CD drive is mounted and the start-up scripts are executed. For Linux systems, this can usually be achieved by adapting the */etc/rc.local* script. For Windows based virtual machines, the context scripts need to be registered as a PowerShell start-up script in the local Group Policy Editor.

In our Linux based images, we then execute a custom script that is located on the virtual CD drive. This script is called *init.sh* and is reading the *context.sh* file that contains the configuration parameters. Then, depending on the use case, further scripts are executed. To illustrate this, an example, the case for our Hadoop cluster images, is explained here.

They are executing two scripts: One to setup networking and one to setup Hadoop. The *net.sh* script is independent of the Hadoop setup and modifies Debian / Ubuntu specific network configuration files in order to use the IP address provided by the context and to setup */etc/hosts* and */etc/hostname* correctly use the host-name provided by the context.

When the network is configured correctly, the Hadoop setup script is run, which first installs SSH keys for the machines to communicate via SSH without the need for passwords. Then, the script decides if it is executed on a master or on a slave node and changes configuration accordingly. After each machine has completed its local setup, it continues to complete the cluster setup. For the master nodes, this means that the master node is waiting for all worker nodes to finish their configuration. The worker nodes are signalling this to the master node via SSH. After all worker nodes have registered with the master node, the master starts up the Hadoop services on all machines and then formats the Hadoop distributed file system.

5. FUTURE WORK

One of the biggest issues with the current middle-ware is that the virtual machine images need to be preloaded on the IaaS platform. The market is not automatically distributing virtual machine images. In further work, we are planning to extend the market system in a way that it is capable of copying disk images, networks and configurations to different clouds.

The current prototype implementation is only tested and running upon OpenNebulas implementation of the OCCI standard. We are planning to extend our market in a way that it also supports other cloud computing infrastructure, like OpenStack or IBM SmartCloud. In order to achieve this, a generic model of virtual machine contextualization has to be found. The currently used virtual machine images are bootable on other infrastructures, but read contextualized configuration only from OpenNebula systems.

In order to successfully support scenarios where one scales out a system to a remote cluster, the system would need extensions in terms of network connection transparency, i.e. allowing the virtual machines of a single user to communicate with each other on a private network, regardless of the IaaS cloud, which an instance of a compute node is launched on. This would allow one to launch different nodes of a single PaaS cluster on different IaaS clouds and bundle resources from different data centers for larger experiments.

For production use of this market system, especially for the education scenario, the need for automatic provisioning of virtual computing instances will arise. Educators should be able to batch instantiate resources for their students, with the possibility to preload additional software and data to the cloud, ready to use for classes and assignments. On the other hand, the system should also be able to automatically shut down unneeded resources, for instance, based on a time-frame, the market place could release all resources that where acquired for a certain class.

Finally, we plan to extend the virtual appliance offerings in our market with new virtual machines covering various areas of cloud computing or distributed computing in general. This will be a basic necessity in order to make our platform successful.

We are planning to evaluate this concept and system along two lines: (1) During a cloud computing lecture planned this fall, students will be able to use both traditional cloud computing stacks, like OpenNebula and OpenStack, or the market application in order to access cloud computing resources. Over the time of the lecture, we will monitor the usage of the different front ends. In a concluding survey, we are planning to evaluate if working with cloud computing resources, which are harnessed by a market application, are easier to use than setting up computing resources manually for each individual project.

(2) On the other hand, we are planning to evaluate the qualitative criteria of our system. Since the market is not influencing the cloud computing infrastructure, only factors inaugurated by the market application are evaluated, e.g. if acquisition or release times are influenced. This could be measured by monitoring the total time of resource acquisition or release from within the market application compared to direct instantiation from a cloud computing stack.

6. CONCLUSION

In this work we have elaborated that by implementing a market targeted to educators and researchers, cloud computing could be facilitated more easily in these areas. Meaning, that by removing the complexity of setting up and config-

uring parts of the cloud computing stack, users are able to concentrate on problem solving rather than on figuring out how their infrastructure or platform works.

A cloud-based market is also capable of using different backing infrastructures. Thus, not only can different PaaS and SaaS systems be used and tested with a single market, the user is also able to use different IaaS systems as needed.

By using standard APIs for accessing infrastructure clouds, we also provide the possibility to scale out to public clouds in order to cope with peak demands in computing resources.

Although the market is not currently thoroughly evaluated, it is made available to the interested public for further investigation. Researchers and students at the Johannes Kepler University Linz are able to use the market for research and education purposes. This test-bed is permanently connected to the universities cloud computing laboratory and is also able to run virtual machines on the cloud infrastructures of our collaborators.

Everyone else is able to download the current source code of the project, as referred to in this paper, and is invited to deploy the market to any Servlet engine.

7. REFERENCES

[1] Amazon Elastic MapReduce. `http://aws.amazon.com/elasticmapreduce/`.

[2] Apache Hadoop. `http://hadoop.apache.org`.

[3] Cloudera Manager. `http://www.cloudera.com/products-services/tools/`.

[4] Hadoop on Demand. `http://hadoop.apache.org/common/docs/r0.17.0/hod.html`.

[5] Storm: Distributed and fault-tolerant realtime computation. `http://storm-project.net`.

[6] R. Campbell, I. Gupta, M. Heath, S. Y. Ko, M. Kozuch, M. Kunze, T. Kwan, L. Lai, H. Y. Lee, M. Lyons, D. Milojicic, D. O'Hallaron, and Y. C. Soh. Opencirrus cloud computing testbed: Federated data centers for open source systems and services research. 2009.

[7] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, Jan. 2008.

[8] R. J. Figueiredo, D. I. Wolinsky, and P. Chuchuaisri. Educational virtual clusters for on-demand mpi/hadoop/condor in futuregrid. 2011.

[9] K. Kambatla, A. Pathak, and H. Pucha. Towards Optimizing Hadoop Provisioning in the Cloud. *Workshop on Hot Topics in Cloud Computing*, 2009.

[10] K. Keahey and T. Freeman. Contextualization: Providing One-Click Virtual Clusters. In *IEEE Fourth International Conference on eScience*, pages 301–308, 2008.

[11] S. Krishnan, M. Tatineni, and C. Baru. myhadoop - hadoop-on-demand on traditional hpc resources. 2011.

[12] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The Eucalyptus Open-source Cloud-computing System. In *9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 124–131. IEEE, 2009.

[13] R. Nyren, A. Edmonds, A. Papaspyrou, and T. Metsch. Open cloud computing interface - core. Technical report, 2011.

[14] B. P. Rimal, E. Choi, and I. Lumb. A taxonomoy and survey of cloud computing systems. In *Fifth International Joint Conference on INC, IMS and IDC*, pages 44–51, 2009.

[15] P. Sempolinski and D. Thain. A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus. In *2nd International Conference on Cloud Computing Technology and Science*, pages 417–426. IEEE, 2010.