2011 Third IEEE International Conference on Coud Computing Technology and Science

Towards Reproducible eScience in the Cloud

Jonathan Klinginsmith [#1], Malika Mahoui [*2], Yuqing Melanie Wu [#3]

[#]*School of Informatics and Computing, Indiana University*
Bloomington, IN, USA
[1]jklingin@indiana.edu
[3]yuqwu@indiana.edu

[*]*School of Informatics, IUPUI*
Indianapolis, IN, USA
[2]mmahoui@iupui.edu

***Abstract*— Whether it be data from ubiquitous devices such as sensors or data generated from telescopes or other laboratory instruments, technology apparent in many scientific disciplines is generating data at rates never witnessed before. Computational scientists are among the many who perform inductive experiments and analyses on these data with the goal of answering scientific questions. These computationally demanding experiments and analyses have become a common occurrence, resulting in a shift in scientific discovery, and thus leading to the term *eScience*.**

To perform eScience experiments and analysis at scale, one must have an infrastructure with enough computing power and storage space. The advent of cloud computing has allowed infrastructures and platforms to be created with theoretical limitless bounds, thus providing an attractive solution to this need.

In this work, we create a reproducible process for the construction of eScience computing environments on top of cloud computing infrastructures. Our solution separates the construction of these environments into two distinct layers: (1) the infrastructure layer and (2) the software layer. We provide results of running our framework on two different computational clusters within two separate cloud computing environments to demonstrate that our framework can facilitate the replication or extension of an eScience experiment.

I. Introduction & Motivation

Data-intensive science is considered part of a relatively new paradigm in the scientific discovery process [1]. The term *eScience* [2] has been coined to help explain the common themes and processes found in this new shift in scientific thought. One common need across all eScience is to have an infrastructure available to be able to store and manage the data as well as have the computational power necessary to process the data into scientific insight.

Cloud computing [3] has emerged in recent years because of advances in virtualization software. Users of cloud computing environments can now take advantage of seemingly endless amount of virtual resources such as computers and storage. By providing a scalable and elastic computing infrastructure, cloud computing is an attractive solution for the demanding experiments run in eScience research disciplines. The cloud has also opened up opportunities for researchers in all of the eScience fields to move experiments to a more common ground, thus enabling greater reproducibility.

As a researcher reading a scientific paper on a new algorithm within a particular eScience domain, it can be challenging to replicate the authors' computationally intensive experiments. To fully reproduce the experiments in the paper, one must have both the infrastructure and software configured in the same manner, as well as have access to the data used within the original experiment. In many cases, having access to all these items is not possible [4]. Even if the original data are not available, it should be reasonable to expect experimental setup to be reproducible. Specifically, if the infrastructure setup and the software installation and configuration can be performed in a reproducible manner then scientists are much more enabled at replicating or extending the experiment in question.

Therefore, in this work, we demonstrate through the use of infrastructure automation and cloud computing the concept of reproducible eScience is an achievable goal. To this end, this work enables eScience researchers to spend less time on the process of recreating a previous experiment and more time on enabling and advancing scientific work. Towards the goal of performing reproducible eScience experiments in the cloud, we demonstrate the following:

- The construction of scalable computing environments into two distinct layers: (1) the infrastructure layer and (2) the software layer.
- A demonstration through this separation of concerns that the installation and configuration operations performed within the software layer can be re-used in separate clouds.
- The creation of two distinct types of computational clusters, utilizing the framework.
- Two fully reproducible eScience experiments built on top of the framework.

II. Background & Related Work

Virtualization technology has opened the door to many advances in computing. A virtual machine (VM) [5] is a running instance of a computer where resources such as memory and central processing units (CPUs) are allocated through virtualization software. A *virtual cluster* (VC) [6] is a set of VMs and any corresponding storage, which operate

978-0-7695-4622-3/11 $26.00 © 2011 IEEE
DOI 10.1109/CloudCom.2011.89

as a whole to create the presence of a single computational entity.

Cloud computing has emerged in recent years because of advances in virtualization software. Companies such as Amazon, Google, and Microsoft provide services to customers for use of the virtual resources owned by them. The definition of cloud computing has taken many forms in the academic community and industry. For this work, we will use the definitions and terms discussed in [3].

Cloud computing refers to the software, platforms, and infrastructure services provided over the Internet as well as the data centers offering these services. The hardware and virtualization software running on top of this hardware is termed a *cloud*. The property of theoretical limitless bound that has gained much of cloud computing's attention.

Providers of cloud computing services offer application programming interfaces (APIs) as means with which customers subscribe and interact. The lowest level of service available with a cloud is termed *Infrastructure as a Service* (IaaS). At this level, users provision virtual resources such as a VM or a virtual block storage device. We consider this the infrastructure layer whereabouts our framework interacts with the cloud.

The use of cloud computing has been discussed as a means to aid in the reproducibility of *in silico* experiments [7]. Our approach to this problem achieves the same outcome of reproducible computational experiments; however, we have made the conscience decision to maintain only a generic machine image within a cloud. Then utilizing a configuration management tool, we build fully configured VMs based through software installation scripts. Through this approach, we are able to use the same software installation and configuration scripts within the software layer in separate clouds. Another reason for using a software automation approach is that when building a VC there are configurations that are needed at time of construction, such as a worker knowing who the master is or alternatively the master knowing all of the available workers.

The ability to quickly provision a VC within a cloud computing environment is useful in many scenarios within eScience experiments. For example, a researcher can build the necessary environment closer to the data. Both [8], [9] discuss the fact that as the size of data sets grow, and if the construction of a VC can be performed in a straightforward manner, then it is possible to move the computing infrastructure to the data. This realization is in contrast to the traditional mechanism of moving the data to the location of a dedicated HPC cluster, such as a research university's supercomputer.

Eucalyptus [10] is an open-source cloud computing toolkit that provides IaaS capabilities. Using the Eucalyptus toolkit, one can create a cloud with a compatible API to that of Amazon's EC2 and S3. During this research work, we both implemented and tested much of our effort on Amazon as well as FutureGrid's Eucalyptus cloud.

There are several configuration management tools available for use in automating software configurations and installations. Tools such as CFEngine [11] and Puppet [12] have been created to manage and configure infrastructures and systems. In this work we have chosen to use Chef [13] as the tool for the construction of our virtual clusters.

III. Design & Implementation

A. Overall Design

The solution we propose in this work is built in two layers. The first layer, which we call the *infrastructure* layer, is the layer that interacts with API provided by a cloud computing IaaS offering. Typical tasks performed at this layer include instantiating a VM; configuring networking ingress rules; and creating and allocating block storage, among other tasks.

The second layer, which we call the *software* layer, deals with interactions on a running VM. This layer has been separated from the infrastructure layer because interactions at this layer can be performed in a repeatable manner regardless of where the VM resides. Therefore, this separation of concerns allows one to utilize a configuration management tool to build and maintain VCs as well as install and configure eScience applications. Figure 1 exhibits how a client machine interacts with these two layers. For the infrastructure layer, the client interacts with the IaaS cloud provider API whereas within the software layer the client is executing commands directly on the running virtual machine.

Fig. 1. Client interactions between the infrastructure and software layers.

To provide context to what is performed within these layers, Figure 2 demonstrates the common steps needed to create a fully functioning running instance of a VM in the IaaS cloud. The layer of interaction (infrastructure or software) needed to complete the tasks is also included in this figure. By recognizing the interactions between these two layers, it is possible to separate concerns, which leads to creating a repeatable installation and configuration processes for items within the software layer.

B. Implementation

To implement our design, we built equivalent machine images within Amazon's EC2 and FutureGrid's Eucalyptus cloud. The following steps were performed in each cloud:

- Instantiated a base CentOS 5.6 machine image (MI).
- Installed the configuration management software Chef (specifically, the `chef-client` application).
- Built a new machine image.
- Registered the new machine image.

Figure 3 displays the final result of the equivalent machine images, called an Amazon Machine Image (AMI) within Amazon EC2 and Eucalyptus Machine Image (EMI) within the Eucalyptus cloud.

The fundamental outcome to the tasks described above is that equivalent base MIs were created in separate IaaS

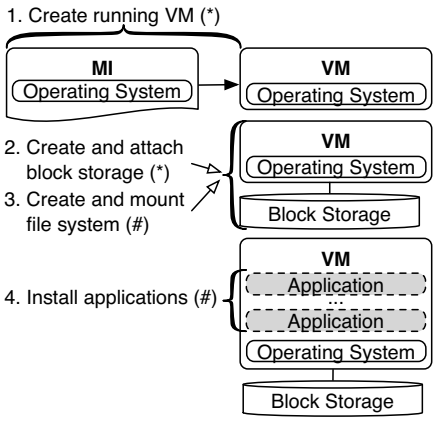

Fig. 2. Example of steps needed to create a running instance with storage and software. * = Infrastructure layer task. # = Software layer task.

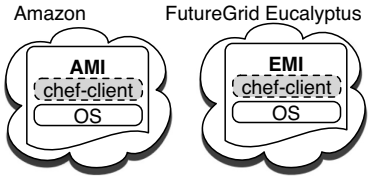

Fig. 3. Equivalent machine images (MI) were built in separate clouds.

environments. From a software installation and configuration perspective, this equivalent infrastructure layer establishes a common underpinning in separate clouds. As a result, one is able to replicate VCs in separate clouds using a common software layer.

C. Chef

Chef [13] is the configuration management tool chosen for automating the construction of our VCs as well as the installation and configuration of software within the clusters. Additional information regarding the Chef architecture is presented online [13]. Below, we provide information on Chef to establish a base understanding of how this configuration management tool is used to construct VCs and reproducible eScience experiments.

- **Cookbook** - A cookbook is a grouping of Chef artifacts (e.g., recipes) typically related to each other.
- **Recipe** - A recipe is the basic unit for configuration in Chef. There can be more than one recipe within a cookbook, and recipes can interact with other Chef recipes and artifacts.
- **Resource** - A resource is used within a recipe to perform an action in an abstracted manner.
- **Chef server** - The central location where all of the Chef artifacts are stored.
- **Chef client** - Machine(s) where the Chef server administers recipes to install and configure software. The client executes the `chef-client` application.
- **Knife** - A command line tool provided with Chef for administrative use. It is possible to create plug-ins into Knife, which was performed in this work. Knife commands are discussed further below, and examples are provided in Tables I and II.

Figure 4 presents the steps necessary to create a virtual cluster in the cloud using our solution. The Chef Server VM, hosting all of our software installation and configuration information, ran within the FutureGrid Eucalyptus cloud.

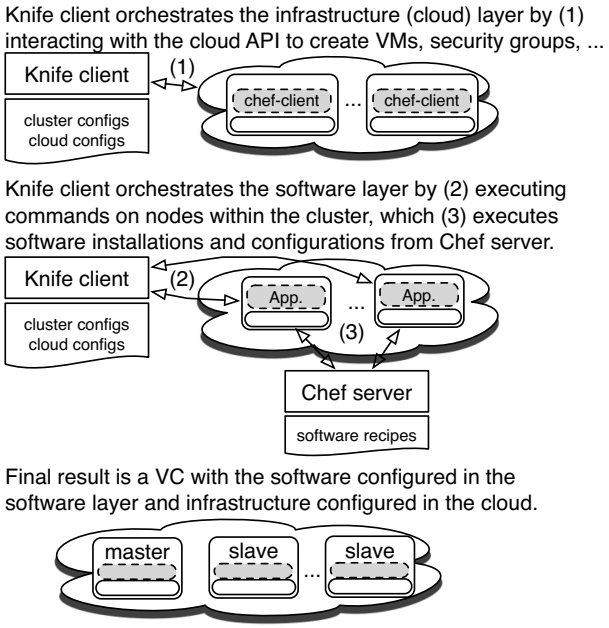

Fig. 4. Overall steps to create a cluster and setup eScience experiment.

IV. Applications & Experiments

A. Hadoop Cluster

Hadoop [14] is an open-source software project sponsored by the Apache Software Foundation [15]. The base contribution of the project is a framework, which implements the MapReduce programming paradigm introduced in [16]. The Hadoop framework provides a distributed, scalable, and reliable mechanism for users to process large amounts of data.

The use of Hadoop and MapReduce has gathered attention in recent eScience work [17], [18], [19] because many eScience algorithms exhibit pleasingly parallel properties [20]. Therefore, the ability to create a Hadoop cluster in a reproducible manner is valuable in itself.

The first step to creating the fully reproducible eScience experiment using Hadoop is to automate the construction of a Hadoop VC. For this process, we modeled our command line interface off of the `hadoop-ec2` command line tool provided in the base Hadoop distribution [21]. We created `hadoop` recipes for performing the set of installation and configuration tasks to create a working Hadoop Cluster.

Within the Knife client machine, we created a plug-in to execute the commands necessary to provision the cloud

```
knife hadoop launch {name} {slave count}
knife hadoop terminate {name}
```

TABLE I
EXAMPLE COMMANDS TO MANAGE A HADOOP CLUSTER.

resources, such as a security groups and VMs, as well as install and configure the Hadoop cluster. Configuration and status information related to the two layers was stored in a database on the client machine. Table I provides example commands executed on the Knife client to launch and terminate a Hadoop cluster, respectively.

After getting a Hadoop cluster working with our design, we extended this capability by automating the installation and testing of the CloudBurst [17] algorithm. CloudBurst is a parallel read-mapping algorithm used to map DNA sequencing data to a reference genome. CloudBurst has been designed to run linearly as the Hadoop cluster scales. The CloudBurst web site provides a sample data set. To execute the CloudBurst sample experiment, the following steps are needed :

1) Download and install CloudBurst
2) Download and extract the sample data
3) Load data into Hadoop's distributed file system (HDFS)
4) Execute the Hadoop run
5) Extract generated results from HDFS
6) Execute `diff` on the generated vs. expected results

To perform these steps in a repeatable manner, we created a `default` receipe within Chef cookbook named `cloudburst`. Once the Hadoop cluster is running, the Hadoop Master node executes the `chef-client` command to perform the CloudBurst experiment steps. By creating these sets of installation and configuration Chef recipes, we were able to perform the CloudBurst experiment by executing the following three steps. Step 1. is run on the Knife client machine. All other steps are run on the Hadoop Master.

1) knife hadoop launch cloudburst 9
2) echo '{"run_list": "recipe[cloudburst]"}' > cloudburst.json
3) chef-client -j cloudburst.json

Step 1 launches the Hadoop cluster with nine slave machines. Step 2 creates a JavaScript Object Notation (JSON) [22] configuration file, which contains a list of items to download and run from the Chef Server. In this case, the steps to perform the CloudBurst sample data experiment. Step 3 runs the Chef `cloudburst` default recipe, which in turns performs all the steps to recreate the entire sample data experiment.

The CloudBurst sample data example executes two separate MapReduce tasks named within the code as *CloudBurst* and *FilterAlignments*. Figure 5 displays the total run-time for these two tasks, executed on the three separate clusters. To reproduce the results displayed in Figure 5, the number of slaves in Step 1 was modified from 9, 19, to 49, which in turn created a Hadoop cluster size 10, 20, and 50, respectively. This experiment confirmed the ability to configure a running Hadoop cluster and run a sample experiment using a handful of commands. It was possible to automate Step 2 and 3, but for sake of clarity we explicitly stated these tasks.

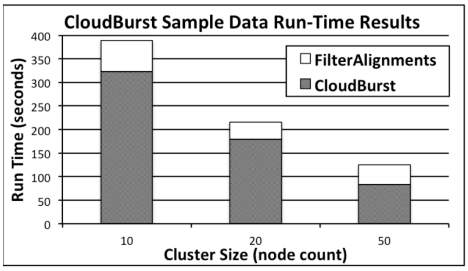

Fig. 5. Run-time results of running CloudBurst sample

B. Condor Pool with a Distributed File System

Condor [23] is a high performance computing workload management system. A Condor pool is collection of machines that process resource requests. Within a pool, there is one machine called a *Central Manager*. The Central Manager collects information on machines within the pool and intermediates the match-making between resources and resource requests. The *Execute* and *Submit* roles are two other possible roles for machines within a Condor pool. Each machine implementing the Execute role will advertise its available resources (CPU, memory, etc.) so that tasks can be executed on it. The Submit role allows a machine to submit Condor jobs within the pool. To create a working Condor pool at least one machine needs to implement each of these roles.

We created a `condor` Chef cookbook that installs Condor on each of the machines in a pool. A node is configured at runtime with one or more of the Condor roles described above, which starts the appropriate Condor daemons. Additionally, the recipes populate Condor configuration files, including setting the Central Manager on each of the Execute machines.

Many IaaS clouds have a limitation where a block storage volume can only be attached to a single running instances. This is true for Amazon EC2 and FutureGrid's Eucalyptus, the two clouds used in this work. Therefore, to have separate nodes within the pool access the same block storage, a distributed file system must be created. For the implementation of the Condor pool, the open source GlusterFS [24] distributed file system was used.

To use GlusterFS, a `gluster` Chef recipes were created for installing and configuring GlusterFS servers and clients. The Condor Central Manager was configured to be a GlusterFS server. Consequently, the Central Manager was the only node in the Condor Pool that had block storage attached to it. All nodes within the pool, including the Central Manager, were configured to be GlusterFS clients. Figure 6 provides a graphical display of the Condor Pool configuration. Further, Table II provides example commands executed on the Knife client to launch, terminate, and add nodes to the Condor pool, respectively.

After building a base Condor pool with a distributed file system, we installed the Pegasus [25] eScience workflow framework. A Chef recipe was created within a `pegagus`

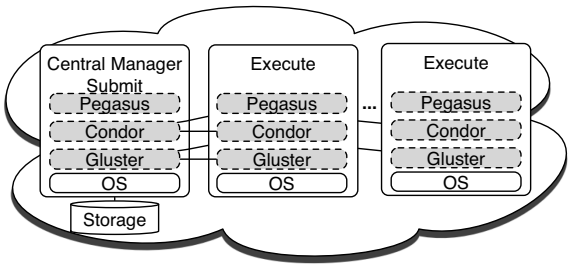

Fig. 6. Condor pool with Pegasus and a Gluster distributed file system.

```
knife cluster launch {name} {exec. host count}
knife cluster terminate {name}
knife cluster node add {name} {node count}
```

TABLE II
EXAMPLE COMMANDS TO MANAGE A CONDOR POOL.

cookbook to install Pegasus on each of the nodes within the Condor pool.

Within the Pegasus distribution, an example workflow named *condor-blackdiamond* is included. The Black Diamond workflow creates a simple directed acyclic graph (DAG) using Condor's Directed Acyclic Graph Manager (DAGMan) meta-scheduler. A `black_diamond` recipe was created in the `pegagus` cookbook to install the *condor-blackdiamond* example within the `/mnt/gluster` directory of the distributed file system. By creating these Chef recipes, we were able to perform the Pegasus Black Diamond example by executing the following steps. Step 1. is run on the Knife client machine. The remaining steps are run on the Condor Central Manager.

1) `knife cluster launch pegasus 1`
2) `echo '{"run_list": "recipe[pegasus::black_diamond]"}' \ > black_diamond.json`
3) `chef-client -j black_diamond.json`
4) `su - cluster`
5) `cd /mnt/gluster/condor-blackdiamond`
6) `./submit condorpool`

V. CONCLUSIONS & FUTURE WORK

We made evident in this work that cloud computing can be used for researchers to reproduce eScience experiments and applications in a straightforward manner. Two distinct types of computational clusters, a Hadoop cluster and a Condor pool, were created utilizing the methodology of separating the infrastructure layer, which interacts with a cloud IaaS API from the the software layer, which interacts with a running virtual machine. Finally, we demonstrated our work by running two fully reproducible eScience examples within our framework.

In the future we plan to extend this work by providing additional configuration and management capabilities to the virtual clusters. These additional capabilities, in turn, provide more flexibility for the creation and management of virtual clusters. Next, is to further investigate ability to have virtual clusters adapt to infrastructure and workflow demands by utilizing cloud APIs and monitoring capabilities.

VI. ACKNOWLEDGEMENT

This paper was developed with support from the National Science Foundation (NSF) under Grant No. 0910812 to Indiana University for "FutureGrid: An Experimental, High–Performance Grid Test–bed." Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

[1] G. Bell, T. Hey, and A. Szalay, "Beyond the data deluge," *Science*, vol. 323, no. 5919, pp. 1297–1298, 2009.
[2] T. Hey, S. Tansley, and K. Tolle, "Jim gray on escience: A transformed scientific method," *Microsoft Research*, 2009.
[3] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," University of California at Berkeley, Tech. Rep. UCB/EECS-2009-28, 2009.
[4] S. Staff, "Challenges and opportunities," *Science*, vol. 331, no. 6018, pp. 692–693, 2011.
[5] R. Goldberg, "Survey of virtual machine research," in *IEEE Computer*, 1974, pp. 34–45.
[6] I. Foster, T. Freeman, K. Keahey, D. Scheftner, B. Sotomayor, and X. Zhang, "Virtual clusters for grid communities," in *Proc. of the 6h IEEE Int. Symp. on Cluster Computing and the Grid*, 2006, pp. 513–520.
[7] J. Dudley and A. Butte, "In silico research in the era of cloud computing," *Nature Biotechnology*, pp. 1181–1185, 2010.
[8] S. Leo, P. Anedda, M. Gaggero, and G. Zanetti, "Using virtual clusters to decouple computation and data management in high throughput analysis applications," in *Proc. of the Euromicro Conference on Parallel, Distributed, and Network-Based Processing*, 2010, pp. 411–415.
[9] P. Anedda, S. Leo, S. Manca, M. Gaggero, and G. Zanetti, "Suspending, migrating and resuming hpc virtual clusters," *Future Generation Computer Systems*, vol. 26, no. 8, pp. 1063 – 1072, 2010.
[10] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *Proc. of the 2009 9th IEEE/ACM Int. Symp. on Cluster Computing and the Grid*, 2009, pp. 124–131.
[11] "Cfengine." [Online]. Available: http://cfengine.com
[12] "Puppet labs." [Online]. Available: http://www.puppetlabs.com/
[13] "Chef." [Online]. Available: http://www.opscode.com/chef/
[14] "Hadoop." [Online]. Available: http://hadoop.apache.org/
[15] "Apache." [Online]. Available: http://www.apache.org
[16] J. Dean and S. Ghemawat, "MapReduce: Simplied data processing on large clusters," in *Commun. ACM*, vol. 51, no. 1, 2008, pp. 107–113.
[17] M. Schatz, "CloudBurst: Highly sensitive short read mapping with MapReduce," *Bioinformatics*, vol. 25, no. 11, pp. 1363–1369, 2009.
[18] C. Zhang, H. De Sterck, A. Aboulnaga, H. Djambazian, and R. Sladek, "Case study of scientific data processing on a cloud using hadoop," in *High Performance Computing Syst. and Applicat.* Springer Berlin / Heidelberg, 2010, vol. 5976, pp. 400–415.
[19] J. Ekanayake, T. Gunarathne, and J. Qiu, "Cloud technologies for bioinformatics applications," *IEEE Trans. on Parallel and Distributed Syst.*, pp. 998–1011, 2011.
[20] T. Gunarathne, T.-L. Wu, J. Qiu, and G. Fox, "Cloud computing paradigms for pleasingly parallel biomedical applications," in *Proc. of the 19th ACM Int. Symp. on High Performance Distributed Computing*, 2010, pp. 460–469.
[21] "Running hadoop on amazon ec2." [Online]. Available: http://wiki.apache.org/hadoop/AmazonEC2
[22] "Json." [Online]. Available: http://www.json.org/
[23] T. Tannenbaum, D. Wright, K. Miller, and M. Livny, "Condor – a distributed job scheduler," in *Beowulf Cluster Computing with Linux*. MIT Press, 2001.
[24] "Glusterfs." [Online]. Available: http://www.gluster.com/products/glusterfs/
[25] E. Deelman, G. Mehta, G. Singh, M.-H. Su, and K. Vahi, "Pegasus: Mapping large-scale workflows to distributed resources," in *Workflows for e-Science*, I. J. Taylor, E. Deelman, D. Gannon, and M. Shields, Eds. Springer London, 2007, pp. 376–394.