

# Wrangler: Virtual Cluster Provisioning for the Cloud

Gideon Juve and Ewa Deelman  
USC Information Sciences Institute  
Marina del Rey, CA 90292 USA  
{gideon,deelman}@isi.edu

## ABSTRACT

Cloud computing systems are becoming an important platform for science applications. Infrastructure as a Service (IaaS) clouds provide the capability to provision virtual machines (VMs) on demand with a specific configuration of hardware resources, but they do not provide functionality for managing those resources once provisioned. In order for such clouds to be used effectively for parallel and distributed scientific applications, tools need to be developed that can help users to deploy their applications in the cloud. This paper describes a system we have developed to provision, configure, and manage clusters of virtual machines.

## Categories and Subject Descriptors

D.4.7 [Operating Systems]: Organization and Design—*distributed systems*.

**General Terms:** Management, Design.

**Keywords:** Cloud computing, virtual clusters.

## 1. INTRODUCTION

Infrastructure as a Service (IaaS) clouds are becoming an important platform for scientific computation. IaaS clouds allow users to provision computational resources in the form of virtual machines (VMs) from commercial and academic resource providers. Unlike other distributed resource sharing solutions, such as grids, users of infrastructure clouds are given full control of the entire software environment in which their applications run. This enables the environment to be customized to suit the specific needs of the application, but it comes at the cost of increased complexity and some additional effort required to setup and deploy the application.

Current infrastructure clouds provide interfaces for allocating individual VMs with a desired configuration of CPU, memory, disk space, etc. However, these interfaces typically do not provide any features to help users configure the execution environment for their application. Distributed scientific applications typically run in cluster environments that have a distributed storage system for sharing data between several nodes, and a resource manager for mapping tasks to nodes. The challenge faced by application developers is how to deploy *virtual clusters* [1] in the cloud given the dynamic nature of cloud environments. Tools are needed that can automatically provision and configure virtual clusters.

Creating a virtual cluster is not a trivial task. It is usually not sufficient to simply develop a virtual machine (VM) image that runs the appropriate services when the virtual machine starts up, and then just deploy that image on several VMs in the cloud. Often the configuration of cluster services requires information about all the

nodes in the cluster (such as IP addresses, host names, etc) as well as parameters specified by the user. This information cannot be hard-coded into an image when it is created because it is not known at that time. Instead, the information must be provided after the resources are provisioned. A user could manually provide this information, but doing so is time consuming and error prone, especially for large virtual clusters. Instead, we advocate an approach where the information is provided by a service to which the user delegates responsibility for provisioning and configuring their virtual cluster.

## 2. WRANGLER

We have developed a system called Wrangler that automatically provisions and configures virtual clusters in the cloud. The system allows users to send a simple XML description of the desired virtual cluster to a web service, which manages the provisioning of virtual machines and the deployment of software and services. Unlike similar solutions [3] it is capable of interfacing with many different cloud resource providers. Currently the system supports Amazon EC2 ([aws.amazon.com](http://aws.amazon.com)), Eucalyptus ([eucalyptus.com](http://eucalyptus.com)), and OpenNebula ([opennebula.org](http://opennebula.org)). We plan on adding additional interfaces in the future.

Virtual clusters are specified using a custom XML format. The XML format describes virtual clusters as a collection of several *nodes*, which correspond to virtual machines. Each node has a *provider* that specifies the cloud resource provider to provision the node from, and defines the characteristics of the virtual machine to be provisioned, such as the VM image to use and the hardware resource type (CPU, memory, disk, etc.). Each node can have multiple *roles*, which describe the functions that will be performed by the node. Each role is associated with a script, called the *role script*, that will be executed on the node to configure it for that role. Roles can be customized using *parameters*, which are passed to the role script when it is executed on the node. Role scripts can be any executable file, but are typically shell, Python or Perl scripts. Users can write their own scripts to implement a custom role.

The architecture of Wrangler is shown in Figure 1. The components of the system include:

**Client**—The client is a command-line program and API that sends requests to the coordinator to create new clusters, list active clusters, terminate clusters, and retrieve detailed cluster status. Any role scripts required by a new cluster are sent by the client to the coordinator, where they can be distributed to the nodes in the cluster. This makes it easy for users to define new roles without creating a new VM image.

**Coordinator**—The coordinator is a service that manages the provisioning of virtual machines from cloud resource providers, assigns roles to virtual machines, and acts as an information broker for all virtual clusters. The coordinator collects information about each VM from the agent, and uses that information to direct the configuration of the virtual cluster.

**Agent**—An agent runs on each of the provisioned virtual machines to manage their configuration and monitor their health. Agents have provider-specific *adapters* that they use to collect information about the VM, such as the IP address, host name, and other dynamic data, which is sent to the coordinator. Upon receiving information from the coordinator about the VM’s roles, the agent retrieves the appropriate role scripts and invokes them to configure the VM.

All of the components of Wrangler were developed in Python and communicate using XML-RPC with SSL. Authentication between components is done using X.509 certificates, which are generated and signed by the coordinator.

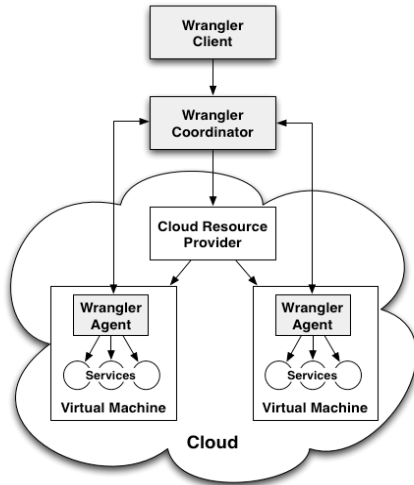


Figure 1: Wrangler architecture

### 3. EVALUATION

In this section we demonstrate the use and performance of Wrangler using an example virtual cluster configuration for workflow applications. The virtual cluster consists of a master node that manages the workflow and stores data, and N worker nodes that execute workflow tasks. The master node is configured with an `nfs_server` role that exports an NFS file system, a `condor_master` role that creates a Condor pool [4], and a `pegasus` role that installs the Pegasus workflow management system [2]. The worker nodes have an `nfs_client` role that mounts the master’s NFS file system, and a `condor_worker` role that configures the node to execute workflow tasks sent by the Condor master.

We ran several experiments on Amazon EC2, NERSC’s Magellan cloud (<http://www.nersc.gov/nusers/systems/magellan>), and FutureGrid’s Sierra cloud ([futuregrid.org](http://futuregrid.org)) to determine the time required to deploy this virtual cluster. The performance of Wrangler is primarily a function of the provisioning time of the underlying cloud platform, which is shown in Table I. Table II shows the time for Wrangler to provision a simple N node virtual cluster without any roles, and Table III shows the time to provision an N node virtual cluster with all of the roles required to execute a workflow application. “FAIL” indicates that the experiment was not completed due to repeated failures during provisioning.

These experiments show that the overhead of Wrangler is not significant. Comparing Tables I and II shows that Wrangler adds only a small overhead to the provisioning time of the underlying cloud platform. This overhead is primarily a result of the fact that Wrangler waits for all N nodes to start before configuring the

cluster, and the fact that the provisioning time of the nodes varies by some amount. Comparing Tables II and III shows that applying roles also only adds a small overhead. This is mostly dependent upon the specific roles required. In this case, the overhead is primarily due to the amount of time required to export and mount the NFS file system.

TABLE I. SINGLE NODE PROVISIONING TIME.

	Min	Max	Mean
Amazon	44.93	64.53	55.40
Magellan	96.06	130.74	104.88
Sierra	337.55	584.92	428.70

TABLE II. PROVISIONING TIME OF A SIMPLE VIRTUAL CLUSTER WITH NO ROLES.

	2 Nodes	4 Nodes	8 Nodes	16 Nodes
Amazon	55.86	55.59	69.88	112.70
Magellan	101.66	102.12	131.64	206.26
Sierra	371.01	455.70	500.85	FAIL

TABLE III. PROVISIONING TIME OF A VIRTUAL CLUSTER FOR WORKFLOW APPLICATIONS.

	2 Nodes	4 Nodes	8 Nodes	16 Nodes
Amazon	101.22	111.22	98.49	112.52
Magellan	173.87	175.07	185.34	349.81
Sierra	447.53	432.95	508.54	FAIL

### 4. CONCLUSION

In this paper we presented the design of a system that automatically provisions and configures virtual clusters in the cloud. We have shown how this system can be used to deploy virtual clusters for running scientific workflow applications in the cloud, and presented a simple performance evaluation that shows that the system adds only a small overhead to the provisioning time.

In the future we plan to investigate solutions for automatically detecting and responding to virtual machine failures, for managing dependencies between nodes in complex virtual cluster configurations, and for dynamically changing the size of a virtual cluster in response to changes in the resource requirements of an application.

### 5. ACKNOWLEDGEMENTS

This work was supported by the National Science Foundation under grant OCI-0943725.

### 6. REFERENCES

- [1] J.S. Chase, et. al., “Dynamic virtual clusters in a grid site manager,” *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC03)*, 2003, pp. 90-100.
- [2] E. Deelman, et. al., “Pegasus: A framework for mapping complex scientific workflows onto distributed systems,” *Scientific Programming*, vol. 13, 2005, pp. 219-237.
- [3] K. Keahey and T. Freeman, “Contextualization: Providing One-Click Virtual Clusters,” *4th International Conference on eScience (eScience '08)*, 2008.
- [4] M.J. Litzkow, M. Livny, and M.W. Mutka, “Condor: A Hunter of Idle Workstations,” *8th International Conference of Distributed Computing Systems*, 1988.