A Scalable Approach to Deploying and Managing Appliances

R. Bradshaw, N. Desai, T. Freeman, and K. Keahey

Abstract— The use of virtualization in Grid computing has seen a lot of interest lately. However, while much effort has been expanded on developing the capabilities of Virtual Machine Monitors (VMMs) and associated tools and services relatively little has been done to investigate the requirements underlying the scalable production, deployment, and management of VM images. At the same time, the clear understanding of requirements and capabilities in this area is critical to creating progress in exploring the applications of virtualization. In this paper, we investigate the issues and propose some of the solutions relevant to this question.

Index Terms— virtualization, configuration management

1 Introduction

One of the main motivations for using virtual machines is to easily and scalably provide on-demand environments -- a VM image can be prepared and finely customized ahead of time and then deployed or taken down in a matter of milliseconds. This in practice significantly expands the set of configurations a site can host as we can now switch between even very complex configurations without seriously impacting the time in which resources are available for computation. In practice however, the need to maintain a large number of VM images – potentially orders of magnitude more than physical resources -- as well as the sheer volume required for the storage of such images would pose a barrier to the scalability of this approach.

Further, the ability to deploy a pre-configured image decouples the (typically long) environment configuration process from the (now short) process of binding environments to resources. A VM image can thus be easily and quickly deployed on any site that happens to have available resources. This enables the emergence of a new provisioning model in which a site does not need to understand in detail all the configurations required by its users [1]. However, it poses other problems. First, although the bulk of configuration can be done ahead of time, a small but critical amount of configuration has to be carried out when an image is deployed. This includes for example assigning network addresses and adjusting the configuration of applications relying on them, providing a host certificates for an image, pointing the appliance at site services, and generally making it aware of its *deployment context*. Further, the site administrator needs to be able to establish trust in an image – verify that the image configuration complies with site policies concerning for example offsite root access or presence of software with known security exploits. Without mechanisms addressing these issues, in practice only site-adapted images will be deployable in practice, with effects opposite to what is desired.

In 2003 Sapuntzakis et al [2, 3] introduced the term *virtual appliance,* to describe an application combined with an environment needed by that application to execute in a virtual machine as well as issues relating to the production and deployment of such appliances. A similar concept of *virtual workspaces* [4] was introduced in the context of binding specific environments to resource allocations in Grid computing. As virtual appliances become increasingly associated with resource allocation and workspaces with virtual machine implementation these concepts begin to merge; within the scope of this paper we will use them interchangeably. In this paper we are building on both efforts to develop mechanisms that would leverage existing protocols and configuration management tools that would allow users to build appliances out of VM images and integrate appliances from multiple providers into deployment fabric in non-invasive ways.

The goal of this paper is to develop a holistic approach that would provide scalable and sustainable ways of managing and deploying virtual workspaces implemented as VM images. We will discuss ways of leveraging existing configuration management tools, expemplified by the Bcfg2 system [5], for VM image lifecycle management that will allow systems staff to deploy robust virtualized resources for their users. We will also describe the process of *contextualization* – integration of a appliance in its deployment context -- and discuss its reference implementation using Bcfg2 and the Workspace Service [1].

The paper is organized as follows. In Section 2 we summarize the requirements and introduce Bcfg2 and the Workspace Service – two systems we are building on. Sections 3 and 4 give an overview of VM management and deployment implementations respectively. We summarize in Section 5.

2 Requirements and Background

The need to generate and manage a pool of configurations potentially vastly exceeding the current pool of physical resources places a renewed emphasis on configuration management. We identify the following requirements in this field:

- *Author1 information*
- *Author 2 information*
- *Author 3 information*

- *Image generation.* While systems and applications are often hard to configure, the process of configuration management has been streamlined and automated simplifying much of the arduous and error-prone work involved in putting a system together – these capabilities can be applied to the VM image generation process. In addition to simply generating images in a reliable and streamlined way, a VM configuration system may be required to generate reliable attestation information about an image guaranteeing certain properties of its configuration. By building on the ability to reliably repeat a configuration process we can also avoid storing large amount of pre-configured images by simply generating the required configuration (or parts of those configurations) on the fly.
- *Image management.* The need to maintain a potentially large fleet of personalized images also requires the ability to automatically discover and apply updates. In particular, the ability to maintain security properties of an image is critical for the resource providers to verify it against site policies when deciding on whether to deploy it. Management functions thus need not only to ensure that updates are applied, but also correctly regenerate and sign any attestation information that may be supplied with the image. Additional testing services may be developed to ensure that patched images can be tested for functionality.
- *Image deployment.* A certain amount of configuration sometimes needs to be done at deployment time. This is because some of the configuration information may be assigned late in the deployment process. This includes all the IP address information, or security certificates assigned to a virtual machine.

The methods discussed in this paper assume that we are operating on bootable images only – this gives us the flexibility to customize the image in ways that, based on our experience to date, are required in a typical VM deployment.

In the rest of this section, we describe two systems that our current work is based on: the Bcfg2 system and the Workspace Service.

2.1 The Bcfg2 System

A Bcfg2 system is composed of a configuration server and clients. The server keeps track of various configuration specifications which include information ranging over the OS specification for the configuration, the packages it contains, and the services it runs. The clients are able to retrieve those specifications and operate on them to achieve the required configuration, update a configuration (apply patches, etc.) or validate and diagnose its state. A Bcfg2 client operates by comparing its state against the configuration obtained from the server – the results of that comparison can trigger a variety of actions ranging from applying various mechanisms to resolve differences between the expected and actual state to simply reporting such differences.

A Bcfg2 server operates on a number of incrementally constructed configuration profiles. A basic Bcfg2 profile may for example include an operating system, various security services, and an application. This basic profile may be further refined by including more packages and applications.

In a typical interaction, a client fist authenticates and authorizes to the server – the server will then either identify the clients profile or allow the client to assert its profile. Once the client's profile has been established, the server presents it with a set of probes collecting local information about the client, such as for example its hardware parameters. The information collected by the probes is then uploaded to the server and allows it to build a configuration for the client customizing the generic specification corresponding to the client's profile. This customized target configuration is then downloaded by the client. Comparing the target configuration with its state, the client can then (1) make changes, (2) report the differences, or (3) communicate with other entities on actions that should be taken. At the end of the interaction the client communicates to the server the description of actions taken, configuration errors, things that still need to be done, etc.

2.2 The Workspace Service

The Workspace Service allows authorized users to dynamically deploy and manage workspaces, implemented as Xen virtual machines [6]. The Workspace Service has been developed within the Globus framework and provides WSRF interfaces to Xen functions: clients can deploy, shutdown, pause, and reactivate VMs. A workspace is deployed based on two types of information: (1) a pointer to an image and meta-data describing deployment (contextualization) information and (2) a resource allocation (memory, CPU share, etc.) that is requested for that image. Once deployed, a client can discover relevant information about the image (such as the assigned IP address), manage the image (e.g. increase the time-to-live of a VM), adjust its resource allocation, or terminate the image.

The Workspace Service provides authentication and authorization mechanisms. Among others, we allow a client to authorize based on virtual organization (VO) role information contained in the VOMS [7] credentials. For VM instances that require host certificates of their own the Workspace Service can provde them. These certificates are generated on startup, after the IP address has been assigned to the appliance, and are currently based on the workspace host certificate.

The Workspace Service allows a client to configure networking for the VM accommodating several flexible options (allocating new network address, bridging existing address, etc.). In particular, a client can request configuring a VM on startup with several different NICs allocating different addresses from different pools (e.g., one public and one private). We provide mechanisms for a site to set aside such address pools for the VMs.

Finally, the Workspace Service allows the client to specify resource allocation (comprising memory allocation and CPU share) to be assigned to a VM and manage that resource allocation during deployment. The Workspace Service also provides a local resource manager that has the

pability to manage a pool of nodes on which edge services are deployed to accommodate the service deployment model (as opposed to a batch deployment model). To use it, the nodes are configured with a very lightweight manager client. In particular, this pool of physical nodes can be used as platform for the deployment of virtual clustes.

3 Appliance Lifecycle Dependencies

We define appliances (or workspaces) as environments that can support specific functionality and are portable, i.e. can be automatically adapted to many different deployment contexts. In this section, we will describe the lifecycle of an appliance and discuss the role of appliance providers and their high-level interdependencies with deployment services; the next section will discuss specific dependencies between them.

3.1 Appliance Lifecycle

In the first state of appliance lifecycle, an appliance is creted by an *appliance provider*. We may have providers dedicated to the creation and maintenance of appliances for specific communities, or virtual organizations (VOs), such as the OSG [8] or an application group within that organization. In addition to the configuration and maintenance, the providers attest to the soundness of appliances.

In the second stage of the lifecycle, the appliance is refined and customized. For example, deriving from the original community appliance, an end-user may want to customize by associating it with specific datasets by mounting additional data partitions (attested by this user) or create additional accounts.

The resulting appliance will then be submitted to a deployment service, such as the Workspace Service, which will make a deployment decision in conformance with resource provider's policies based on the provenance of the workspace as well as its attested properties (ranging from assertions on root access to the workspace to its state with respect to security) and provision resources for the appliance.

On deployment, the appliance will be *contextualized* to put it in a specific deployment context, and producing an *appliance instance* that is ready for use. As time passes and the configuration policies of various sites change, e.g. in response to known security exploits, the appliance can be updated to conform to such policies and maintain its deployment range.

3.2 Appliance Management and Deployment

One of the major costs of virtualization is the initial cost of appliance creation, and eventually also of storing many appliances with similar or redundant configuration. Hence the need for appliance providers enabling a cookbook-style approach to configuration construction to make configuring new appliances easier for the end user. The viability of this model has already been illustrated by rPath [9] in development of appliances for high energy physics applications.

As appliances are shared between various sites, trust in an appliance configuration becomes an issue. The workspace service currently provides mechanisms allowing a site to admit applians only coming from a particular VO or application group, but more is needed in practice for an appliance to be admitted to a site – an appliance may be admitted only if it was configured with a given set of security requirements (i.e., only trusted entities have root access to the appliance) and recently patched. These can be generated again based on automatic configuration management tools. As the appliance is initially created, attestation information, including configuration parameters, appliance creator, modification times and appliance origin can be attached in the form of assertions referring to specific software packages as well as to the appliance as a whole. As appliances are modified, fine-grained attestation information will need to be augmented to reflect subsequent reconfiguration operations.

While the Workspace Service already implements methods for authorizing VM deployment requests coming from specific clients or communities (through the use of VOMS credentials or SAML attributes), fine-grained methods are required to process full information about the image itself. Here, we are planning to use the attestation information generated or updated by the appliance generation tools and signed to the image. In [10] we have prototyped the ways of integrating the processing of such information into the appliance deployment process. It is critical however, that this information itself comes from a trusted source and preferably be automatically generated in sync with configuration information on the imge – in practice attestation information coming even from a trusted source may not be as accurate (and therefore as trustworthy) as information generated by trusted software running on a trusted platform.

A persuasive case for automatic management of appliances is made in [3]: appliances need periodic maintenance, much as classical unix systems do. The ability to selectively apply updates, allow the execution of maintenance tasks and perform a periodic re-attestation of an appliance's soundness is of critical importance. Seamlessly performing updates implies the ability to test appliances after maintenance operations have been performed and thus allowing appliance functionality to be validated prior to redeployment minimizing surprises for end users. This requires a test suite attached to an appliance capable of providing such maintenance – in practice such test suite would have to be agreed upon between the appliance provider and the user.

4 Contextualization on Deployment

Appliances cannot be shipped with ready-to-run configurations, because several aspects of configuration are either site dependent (*site binding*) or depend on the particulars of a given instantiation of a virtual appliance (*instance binding*): IP addresses need to be assigned, an appliance may need to be pointed at existing services such as DNS, a batch system head node, or a database. We will call this process *contextualization* as it makes the appliance aware of its deployment context.

To illustrate the need for various steps of contextualization, let's first consider what is required for manual deployment of a 5 node cluster configured with Torque [11] (one headnode and four worker nodes). Typically, we'd

begin the process by assigning required IP addresses to the nodes. We'd then create accounts, point the nodes at a DNS server, configure ssh/scp keys for the cluster nodes, as well as modify Torque configuration files (both depend on IP addres assignment). Additionally, if the headnode is Grid-enabled we might equip it with a host certificate as described in [12], and modify the configuration relevant to the installed Grid services. We will now describe the design and tools required to automate this process.

4.1 Preparing Virtual Appliances for Deployment

As defined in section 3, an essential feature of an appliance is that it can be deployed in many different contexts. Thus, in order turn for example a VM image into an appliance a virtual appliance provider needs to adapt an environment (e.g., represented as a VM image) to consume configuration parameters that are only available after an appliance has been deployed in a specific context. We note that such configuration process consists of two components: (a) configuration parameters specific to a deployment context and (b) a set of environment-specific non-declarative actions that integrate these parameters into a specific environment. We therefore define a virtual appliance as an environment that (a) provides a definition of the contextualization information it requires (*contextualization template*) and (b) contains mechanisms capable of integrating this information into the appliance (*contextualization agent*).

```
<Parameters>
  <Param name='DNSServer'>
    <List>
      <Item value='192.168.1.2'/>
      <Item value='192.168.1.3'/>
    </List>
  </Param>
  <Param name='nodenames'>
    <List>
      <Item value='192.168.7.1'/>
      <Item value='192.168.7.2'/>
      <Item value='192.168.7.3'/>
      <Item value='192.168.7.4'/>
    </List>
  </Param>
</Parameters>

<Param name='users'>
    <List>
      <Item name='user1' value='sad8hgewjnb'/>
      <Item name='user2' value='saasd2sjnb'/>
    </List>
  </Param>
</Parameters>
```

Figure 1: Contextualization information for the Bcfg2 example.

As an example, here are the actions required to turn our Torque VM image into an appliance using the Bcfg2 system as the contextualization agent. The contextualization template for this example (instantiated with information) is shown in Figure 1. The contextualization agent that consumes this information inside the appliance is implemented by Bcfg2 -- a natural choice as it is supplied out of the box with many Linux distributions and implements all of the commands needed to integrate configurations inside of virtual appliances. The contextualization process simply invokes the Bcfg2 agent included in the appliance to consume the parameters provided by the deployment service. The Bcfg2 agent is self-contained, and configured to provide the goal configuration states, taking the supplied parameters into consideration. The appliance creator has included a series of templates that render these parameters into appliance configuration directives.

4.2 Contextualization

On deployment, a deployment service needs to be able to interpret the contextualization template file and provide the expected information values for the required fields. In doing so, the deployment service may need to work with mechanisms that generate the required information, such as a certificate authority generating certificates for appliances. We distinguish between *site binding* parameters (generated by the deployment service once for every site, such as available networks) and *instance binding* parameters (generated for every deployed appliance instance). Once the information is compiled, the deployment service produces a parameter file – a template file associated with the required contextualization information.

After the parameter file is assembled, the information needs to be made available (delivered) to the appliance. In the case of VM images, one option is to use kernel parameters to pass this information – however, only a limited amount of information can be passed in this way, and also in standard Linux, kernel parameters are world readable after the image is deployed making this method unsuitable for passing sensitive information. A more flexible option (and generic across different operating systems) is for the deployment service to simply "patch" the appliance by mounting the VM disk partition and copying the parameter file where it will be expected by the contextualization agent. While this can be time-consuming it is currently the most generic and flexible option and is used by the Workspace Service uses it as the contextualization delivery method.

In our Bcfg2 adaptation, the Bcfg2 agent consumes the parameters supplied by the Workspace service as a file via such an appliance patch. In the example of a Torque appliance, the Bcfg2 agent will first render the list of node names into file contents for "/etc/resolv.conf" to configure domain name resolution. Once this is finished, name resolution functions properly on the appliance for all of the relevant hostnames. Next, the first node in the allocation will be identified as the head node (Bcfg convention), which runs the Torque server and configured to know which resources it controls (the data will be derived from the parameter "nodenames" in Figure 2). The values in this list are included in the Torque configuration. Each client node places the server IP address in the Torque configuration file. At this point, Torque is operational. Bcfg2 is natively capable of setting up ssh keys and trust between nodes and this step is completed next. Finally, user access is configured. Entries for each user included in the users parameter are included in "/etc/passwd". This allows users login access and the ability to execute programs. Once these facets are

configured, the appliance is fully contextualized and operational.

4.3 Generalizing the Approach

In the previous sections we described how a generic VM image can be turned into an appliance by providing contextualization information and a contextualization agent. The question now arises how invasive this process is and to what extent it can be made transparent and easy to use to an appliance developer and user.

First, we note that contextualization agents are already present in many appliances as part of standard system tools and can be leveraged as long as the corresponding contextualization information template can be specified, and compatible delivery methods implemented. The standard DHCP broadcast call which is a part of a typical boot sequence is one such agent. The Workspace Service provides the contextualization information template as part of the workspace meta-data information [1]. In addition the Workspace Service suite of tools also provides a DHCP delivery tool executing on a hypervisor nodes and responding to a VM's boot-up sequence DHCP broadcast. The Workspace Service can dynamically manage DHCP policies assigned to this delivery tool to give a defined set of network configurations to a specific MAC address; MAC address spoofing is controlled, so only the intended information will make it into the image. Further, there is a daemon on each hypervisor node so that the DHCP broadcasts do not escape on to the physical network and interfere with a site's pre-deployed DHCP server's operations. The advantage of using this mechanism is that it allows users to deploy simple appliances with practically no appliance adaptation because DHCP client libraries are virtually ubiquitous and easy to configure in the appliance.

In practice, the need to cover multiple (and often rare) application-specific configurations requires a more flexible if heavier-weight approach. In those cases, existing configuration tools which are often included as part of guest operating system distribution provide a convenient solution – leveraging them facilitates the adaptation process as it eliminates the need to install additional software. In this paper, we used the example of Bcfg2 included with multiple versions of Linux: the user had to configure a Bcfg profile as part of the adaptation process, but the resulting actions were carried out automatically.

5 Related Work

The term *virtual appliance* was introduced by Sapuntzakis et al [2, 3] and their work describes the first attempts at defining contextualization information as well as explaining the requirements for appliance management. We build on this work and extend it to generalize the method and enable the use of generic tools and protocols for configuration management.

rPath [9] is a company that creates software appliances, including virtual appliances compatible with popular VMMs via the rBuilder service. They provide methods of configuring and maintaining appliances; we are currently collaborating with them on defining contextualization services.

Zeroconf [13] is a collection of techniques for establishing networking configurations and service discovery without a centralized information systems. While useful, in the context of dynamically deployed appliances on the grid, the particular networking and service configurations are virtually always either subject to the policies of the deployer and resource provider (and intermediaries if they exist) or desired to be published externally from the VM by the infrastructure.

Configuration management tools such as LCFG [14], Quattor [15] and Smartfrog [16] are somewhat similar in nature to Bcfg2 and could also potentially be used as contextualization agnets. Bcfg2 is our reference implementation of choice due to its availability and convinience.

6 Summary

This paper describes the requirements and services required to ensure the scalable management and deployment of appliances implemented as VM images. We make the case for developing methods of automatic generation and maintenance of VM images – important to achieve scalability as well as develop methods to manage trust and enable more images to be deployed on more platforms in a secure manner taking into account site policies. The issue of establishing methods for trust management between VM image developers and sites supporting their deployment is particularly critical at this stage of technology adoption – applying automatic configuration methods and ensuring a system of timely management and updates will go a long way towards resolving it.

Finally, we described methods of adapting VM images to produce appliances and contextualizing such appliances on deployment. Our preliminary investigation indicates that it is critical for those methods to be lightweight yet flexible to cover the range of required specifications. We also point out that in practice the appliance producer and the appliance deployer have to collaborate on the formatting of the contextualization template: the provider to correctly integrate contextualization information into the appliance and the deployer to correctly interpret them when supplying contextualization information.

In future work we hope to extend the methods described here to provide recontextualization for appliances on migration. Another topic we are investigating is the deployment-time composition of appliances out of components supplied by different parties.

Acknowledgements

We are grateful to Brett Adams, Frank Siebenlist, Ed Smith, Ravi Subramaniam, and Marty Wesley for interesting discussions of the concepts described here. This work was supported by NSF CSR award #527448 and in part, by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, SciDAC Program, Office of Science, U.S. Department of Energy, under Contract W-31-109-ENG-38.

REFERENCES

1. Keahey, K., et al., Virtual Workspaces: Achieving Quality of Service and Quality of Life in the Grid. Scientific Progamming Journal, 2005.
2. Sapuntzakis, C., et al. Virtual Appliances for Deploying and Maintaining Software. in Proceedings of the 17th Large Installation Systems Administration Conference (LISA '03). 2003.
3. Sapuntzakis, C. and M.S. Lam. Virtual Appliance in the Collective: A Road to Hassle-free Computing. in 9th Workshop on Hot Topics in Operating Systems. 2003.
4. Keahey, K., et al. Virtual Workspaces in the Grid. in Europar. 2005. Lisbon, Portugal.
5. Desai, N., et al. BCFG: A Configuration Management Tool for Heterogeneous Environments. in IEEE International Conference on Cluster Computing (CLUSTER'03). 2003.
6. Barham, P., et al. Xen and the Art of Virtualization. in ACM Symposium on Operating Systems Principles (SOSP).
7. The Virtual Organization Management System: http://infnforge.cnaf.infn.it/projects/voms.
8. Open Science Grid (OSG). 2004: www.opensciencegrid.org.
9. rPath: www.rPath.com.
10. Lu, W., et al., Making your workspace secure: establishing trust with VMs in the Grid. SC05 Poster Presentation, 2005.
11. Torque: http://www.clusterresources.com/pages/products/torque-resource-manager.php.
12. Freeman, T., et al., Division of Labor: Tools for Growth and Scalability of the Grids. ICSOC 2006
13. Zeroconf. www.zeroconf.org
14. LCFG: http://www.lcfg.org.
15. Quattor: http://cern.ch/quattor.
16. SmartFrog: http://www.hpl.hp.com/research/smartfrog/.