2008 IEEE International Conference on Services Computing

Simplifying Service Deployment with Virtual Appliances

Changhua Sun*, Le He†, Qingbo Wang† and Ruth Willenborg‡
*Department of Computer Science and Technology
Tsinghua University, Beijing, 100084, China
sunchanghua@tsinghua.org.cn
†IBM China Research Laboratory, Beijing, 100094, China
{lehe,wangqbo}@cn.ibm.com
‡IBM Software Group, Durham, NC, 27703-9141, USA
rewillen@us.ibm.com

Abstract

As IT services become more powerful and complex, service deployment gets more difficult and expensive. Service deployment, the process of making a service ready for use, often includes deploying multiple, interrelated software components into heterogeneous environments. Different technologies and tools try to address these complexities by describing the environments, abstracting the dependencies, and automating the process. Virtual Appliances, a set of virtual machines including optimized operating systems, pre-built, pre-configured, ready-to-run applications and embedded appliance specific components, are emerging as a breakthrough technology to solve the complexities of service deployment. Virtual appliances provide a simple, unified and easy to use interface for service deployment by encapsulating entire custom environments, and resolving the execution policy constraints and inter-dependencies through pre-installing the software applications.

The motivation of this paper is to prove virtual appliances offer a better service deployment mechanism. We start with an easy to understand model to describe the complexity of service deployment and introduce the architecture of a virtual appliance. We then analyze the deployment process of using traditional deployment mechanisms, and quantitatively and qualitatively compare the deployment time, operations and parameters of the traditional approach with the use of virtual appliances. The results show virtual appliances offer significant advantages for service deployment by making the deployment process much simpler and easier, even for the deployment of advanced enterprise services.

1. Introduction

An IT service can be defined as several software components that provide useful functions and can be deployed in a computing environment and composed into overall computing systems or single applications [1], [2]. Service deployment is the process to make the service ready for use. A general deployment process may consist of several interrelated activities including the *release* of service at the end of the development cycle; the *installation* and *configuration* of the service into the execution environment, and the *activation* of the service [3], [4]. Post-installation activities include the *deactivation*, *updating*, *reconfiguration*, *adaptation*, *redeploying* and *deinstallation* of the service [5]. According to an IDC survey, the deployment represents 19% of the estimated $95 billion in total software operation cost [6]. Reducing the cost of service deployment can greatly reduce the whole TCO [7].

Today, IT services are becoming more and more powerful and complex. This is especially true for enterprise services which include many complex and advanced functions. The complexity of service deployment lies in the following three aspects:

- Heterogeneous, distributed environments;
- Execution policies and dependencies;
- Operation coordination and constraints.

The increasing complexity of deployment is mainly due to the heterogeneous environments that the service is deployed into and the service's policies and dependencies. Software applications rely on the availability of specific system resource and libraries. In addition, most of today's applications, especially those for enterprise use, are not standalone systems. They often consist of a large number of components each

978-0-7695-3283-7/08 $25.00 © 2008 IEEE
DOI 10.1109/SCC.2008.53

offering and requiring services of other components. Different components may come from different producers and use different services at the same time. These components have coordination and constraints with each other. Moreover, deployment in distributed, heterogeneous environments adds to the complexity of service deployment.

Many technologies and tools are designed to support service deployment and reduce these complexities by providing descriptions of the environments, abstracting the dependencies and automating the software installation. However, these existing deployment mechanisms have some limitations and do not fully eliminate the complexity of the deployment [5]. For example, RPM Package Manager is limited to a specific operating system and also fails to explicitly model all dependencies.

An approach to avoid the complexities is to obviate these problems entirely by creating perfect custom environments which software applications may be installed into. This can be achieved by **Virtual Appliances**, which are a set of virtual machines with prebuilt, pre-configured, ready-to-run applications packaged along with optimized operating systems [8]. Virtual appliances are emerging as a breakthrough technology, even for enterprise solutions [9] and offer an improved approach for service deployment compared with previous approaches. Virtual appliances encapsulate entire custom environments, and resolve the execution policy constraints and inter-dependencies by pre-installing the software application in a virtual machine image designed to run under VMware [10], Xen [11] or other Virtual Machine Monitors (VMM). Service deployment for or by customers with virtual appliances can be achieved easily by performing some configurations and activating the virtual appliance images.

In this paper, we provide analysis and models for service deployment. We also propose a general architecture for a virtual appliance; analyze the deployment process and provide quantitative and qualitative comparisons against traditional deployment mechanisms in terms of the deployment time, operations and parameters. The goal of the paper is to present and analyze results to determine whether virtual appliances actually offer a more competitive approach for service deployment or not. From the experiments and analysis of the deployment process of using traditional deployment mechanisms and virtual appliances, we prove that virtual appliances offer an improved approach for service deployment, especially for the deployment of advanced enterprise services.

The remainder of the paper is organized as follows. Section 2 describes the related works and discusses the difference with our work. Section 3 gives the basic analysis and models of service deployment. Section 4 explains the concept of a virtual appliance and proposes the architecture and the deployment process of using virtual appliances. Section 5 compares virtual appliances with traditional service deployment mechanisms in terms of the deployment time, operations and parameters. Finally in section 6, we conclude the paper.

2. Related Works

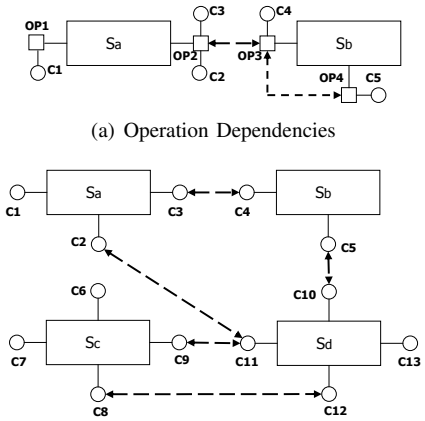
Many tools such as package management system like Red Hat RPM Package Manager and FreeBSD Ports System, Windows Installer [12] and Java platform tools like Enterprise JavaBeans (EJB) [13] are designed and widely used to support service deployment. Carzaniga et al. [3] described the characteristic of software deployment and discussed some technologies at that time. Talwar et al. [1], [2] compared manual, script, language and model based service deployment solutions. These solutions are all considered as traditional service deployment mechanisms in this paper. These tools and technologies do not resolve the complexities of service deployment completely due to the limitations in resolving dependencies among service components and lack of support for heterogeneous environments.

Dearle [5] studied six cases of software deployment technologies and gave some of the future directions. Dearle argued virtualization is likely to have a large impact on software deployment by qualitative statements. The benefits of virtualization and virtual machine are also discussed in [14]–[16]. Chen et al. [14] described how three services, secure logging, intrusion prevention and detection, and environment migration, can take advantage of virtual machines. Wlodarz [15] made a survey of virtualization technologies and listed virtual machines as suitable for hardware replacement, testing and debugging, education and e-Learning, and security systems. Sapuntzakis et al. [17], [18] proposed Collective, a compute utility using virtual appliances to manage systems. Mastrianni et al. [19] also proposed the idea of using appliances to simplify the deployment and management of SMB services.

Our work is to discuss whether virtual appliances offer a more competitive approach for service deployment by providing both quantitative and qualitative data. These related works only provided qualitative statements for some of the advantages.

Table 1. A service deployment sample

Steps	Operations	Remarks
Prepare	Learn all the dependencies	Mandatory
Pre-installation	Adjust OS	Optional
	Adjust environments	Optional
	Remove conflicts	Optional
Installation	Unpack installation	Optional
	Compile and Build	Optional
	Copy files	Mandatory
	Patch	Optional
	Remove dependencies	Optional
Post-installation	Verify the installation	Optional
	Configuration	Optional
	Start the software system	Mandatory



(a) Operation Dependencies

(b) Parameter Dependencies

Figure 1. Service deployment models.

3. Analyzing Service Deployment

In this section, we first analyze the general deployment process of using traditional service deployment mechanisms. Then we define a set of simple models to describe the complexity in service deployment.

3.1. Deployment Mechanisms

Service deployment is the process to make the service, or software system ready for use. Many technologies and tools exist to abstract the environments and dependencies, and to automate the deployment. Though the traditional tools have different methodologies, they share the same general deployment process. For example, they all need to resolve the environment dependencies and the dependencies among the components.

Table 1 shows the general deployment operations for a service in a standalone or a single node environment. Some of the operations are automated by available technologies and tools. These traditional deployment operations can be classified as follows:

1) Learn dependencies: including dependencies with platforms and environments, and dependencies among different software components;
2) Prepare for the installation: including resolving dependencies with platform and environments;
3) Install: including either building from source files or just copying the required files to the destination directories;
4) Remove inter-dependencies: including resolving dependencies among different software components;
5) Start the software system.

Sometimes, the software system is patched during or after the installation process to update or fix problems. Patch may first backup existing files and configurations, then modify or replace files, and add or remove files. Optionally, patch may rebuild the software system and need verification which adds significant complexity to the deployment.

Deploying services in distributed environments increases complexity. Let's examine the complexity involved in a typical, three-tier J2EE solution with Web Servers, Application Servers and the Database Server. The application servers may be deployed across multiple nodes to form a cluster for high availability and load balancing. A sample deployment process is as follows:

1) Deploy the different software to the nodes, including management, load balancing, and application servers;
2) Establish the relationships between the management nodes and the application servers;
3) Deploy application services;
4) Enable load balancing across the nodes;
5) Establish relationships with the first tier web server(s);
6) Establish relationships to the third tier database server.

3.2. Deployment Models

IT system to provide services can be modeled as consisting of several software components. A single component is a minimum indivisible entity in service deployment. Fig. 1 gives an abstract illustration of service deployment using traditional deployment mechanisms. Fig. 1(a) shows the deployment of two components. The deployment may include several operations, and each operation may need some configurations or parameters. **S** represents a software component, (an operating system is also considered as a component

in service deployment); **c** represents a configuration or parameter; and **op** represents an operation to perform configurations. For example, **Sa** needs two operations, and **op1** needs to configure parameter **c1**. Fig. 1(a) also depicts the deployment order and dependencies of operations. **op4** should be after **op3** and **op3** after **op2**. The operations of each software component are not shown in Fig. 1(b) as we want to focus on the relationship among the components. In Fig. 1(b), the relationship between the two software components is represented in the form of configuration dependencies. It also indicates there is a deployment order for these components.

To illustrate the difficulties of service deployment, we devise models based on deployment operations and parameters.

Suppose one approach needs m operations to deploy a service. Each operation has a probability to fail. Let the failure probability of operation i be $a_i(i = 1, \cdots, m)$. The whole successful probability rate to deploy the service would be:

$$R_1 = \prod_{i=1}^{m}(1 - a_i). \tag{1}$$

For each operation, we need to configure some parameters. There is a probability of fail for each configuration parameter. Suppose operation i has m_i parameters to configure. And the jth parameter may fail with probability p_j. The failure probability a_i would be:

$$a_i = 1 - \prod_{j=1}^{m_i}(1 - p_j). \tag{2}$$

From (1) and (2), we can get the whole successful probability rate to deploy the service with this approach as:

$$R_1 = \prod_{i=1}^{m}\prod_{j=1}^{m_i}(1 - p_j). \tag{3}$$

For another approach to deploy the service, suppose it needs n operations and the probability of failure operation i is $b_i(i = 1, \cdots, n)$. Similar to the first approach, let operation i have n_i parameters to configure. The jth parameter may fail with probability q_j. The whole successful probability rate to deploy the service would be:

$$R_2 = \prod_{i=1}^{n}(1 - b_i) = \prod_{i=1}^{n}\prod_{j=1}^{n_i}(1 - q_j). \tag{4}$$

To compare these two approaches, we just need to compare R_1 with R_2. If $R_1 < R_2$, we can conclude the second approach reduces the complexity of the service deployment compared to the first one.

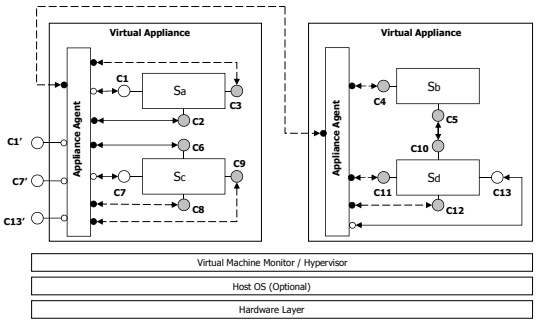


Figure 2. The architecture of virtual appliance.

Suppose $a_1 = a_2 = \cdots = a_m = a$ and $b_1 = b_2 = \cdots = b_n = b$, for the special cases, if we want to satisfy $R_1 < R_2$, we get:

$$(1 - a)^m < (1 - b)^n. \tag{5}$$

and $0 < a, b < 1, m, n \geq 1$.

This equation is always satisfied if $a > b$ and $m > n$. This means we can increase the successful probability rate to deploy the service by decreasing the number of deployment operations and the probability of failure of each operation.

For a general service, if we decrease the number of deployment operations and the failure probability of each operation, we increase the success rate to deploy the service. As each operation is related to the configuration of parameters, by decreasing the number of parameters and their failure probability of configuration, we would increase the success rate to deploy the service, and reduce the complexity of service deployment.

4. Leveraging Virtual Appliances

In this section, we first discuss what a virtual appliance is and then describe the deployment and activation process of using the virtual appliances.

4.1. Basic Architecture

As discussed in Section 1, a virtual appliance is a pre-built, pre-configured, ready-to-run enterprise application packaged along with an optimized operating system inside a virtual machine. To simplify the deployment and provide a user-friendly deployment interface, a general architecture of virtual appliance is proposed as shown in Fig. 2. It has a minimalist virtual machine designed to run under a specific hypervisor, such as VMware or Xen and includes a single application, or a suite of applications to provide a service.

In general, the creation phase of virtual appliance images [20] contains the following steps: 1) Installing a just enough guest operating system; 2) Installing the specific software, such as middleware, database server or applications; and 3) Providing an interface or mechanism for customers to configure the virtual appliance. There are three aspects for the virtual appliance configurations:

- **Guest OS Configuration**: To make the virtual appliances work in different environments, the virtual hardware resources, especially the network parameters, may require changing. Also, most customers want to set their own passwords instead of using default passwords;
- **Software Configuration**: Since the software applications were pre-installed and pre-configured, the changes to the guest OS configuration (network parameters or security) may require application configuration change. Additionally, an appliance may choose to offer some run-time configuration for options that are not pre-configured;
- **Appliance Agent Configuration**: To coordinate, the guest OS and application configuration, an intelligent appliance agent is embedded inside the virtual appliance.

To make the virtual appliance workable in the deployment environment, usually just a few parameters, especially network parameters, are changed and reconfigured. Traditionally, as explained in [20] from VMware, customers can use DHCP or login the virtual appliance and manually configure the network parameters. After the network works, VMware recommends a web interface should be provided for environment and software configurations. Instead of just having a web interface for configuration, our virtual appliance architecture includes a powerful appliance agent to automate the activation of a set of virtual appliances as shown in Fig. 2. The appliance agents are designed to configure the virtual appliances at the first run in a new environment [21], [22]. They can configure the deployment order of software components and resolve the dependencies of parameters among the components.

As shown in Fig. 2, some configurations, like **Sc**'s **c6**, are pre-configured in the creation phase of virtual appliances. Some configurations, like **Sa**'s **c1**, need to be configured by the appliance agent according to the activation time parameters provided by customers. Other configurations, like **Sa**'s **c3**, are related to other virtual appliances' configurations. These are configured at the first run by the related virtual appliances' appliance agents. The agents work together to make sure the dependency configurations are right and consistent. In the following section, we discuss the deployment and activation of virtual appliances.

4.2. Deployment and Activation

After a virtual appliance is created, it encapsulates entire custom environments, and resolves the execution policy constraints and inter-dependencies among the software components. The virtual appliance is ready to be deployed in various environments. During the deployment, there are configurations both inside and outside the virtual appliance. The interface between the hypervisor and the virtual appliance provides a uniform method for configuring virtual hardware resource. For example, this interface is used for configuring the number of virtual CPUs, memory size, virtual disk size, and network parameters.

The appliance agent is leveraged for the configuration inside the virtual appliances. The agent reconfigures the guest OS and applications which were installed during the creation phase according to the requirements of the customer environment. The configuration typically includes the network parameters, such as the DHCP server or static IP and network mask, DNS server, as well as parameters specific to the software applications.

In support of the agent, the parameters are made visible to the virtual appliance. For example, the implementation may use a file, the *Activation Profiles* in [22], to record the parameters provided by customers to automate the deployment and activation. Some tools provided by the virtual machine monitor, for example, a virtual floppy disk (or v-floppy) containing the profiles, can be created. When the v-floppy is attached to the virtual machine, it is automatically visible to the appliance agent. Then, the appliance agent will automate the configuration and activation of virtual appliances according to the parameter files on the first boot [22].

As shown in section 3.1, traditional mechanisms for service deployment in distributed environments are very complex. We simplify the deployment using virtual appliances. To make the deployment easier, three important aspects are considered:

1) Each virtual appliance should be configured and activated;
2) There is an activation order for these virtual appliances;
3) There are configuration dependencies between these virtual appliances.

For example, when DB2 and WebSphere Application server (WAS) are both involved in one ser-

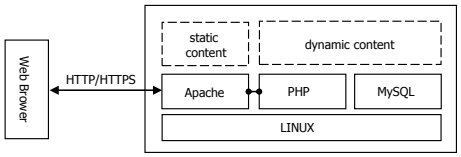


Figure 3. The topology of LAMP.

```
LAMP
1. Prerequisites: like ANSI-C compiler,
   libxm12, etc.;
2. Install apache;
3. Install and configure MySQL;
4. Install PHP at least with:
        --with-apxs2
        --with-mysql
   to support Apache and MySQL;
5. Configure http.conf to add PHP support.
```

Figure 4. Deploying LAMP from source files.

vice, the parameter of DB2: `dbuser.username` should be the same as the parameter of WAS: `jdbc.dbuser.username`. In this example, the appliance agents first check the configuration dependencies and then activate these virtual appliances in the right order. The activation order is predefined in the creation phase. After configuring each single virtual appliance, appliance agents collaborate to configure the dependencies among these virtual appliances. When they finish the configurations, the service is ready for use.

5. Evaluations

In this section, we first give two deployment scenarios. One is a simple and widely used open source solution, and the other is a powerful enterprise solution. For both examples, we compare the virtual appliance deployment scenarios with the traditional service deployment mechanisms.

5.1. Deployment Scenarios

Our first example deployment scenario is a single node with a LAMP (Linux+Apache+MySQL+PHP) stack, whose topology is shown in Fig. 3. Fig. 4 illustrates the traditional deployment steps for a LAMP stack using source files. First, we install PHP support, Apache and MySQL. Additionally, we need to configure the Apache support for PHP. By contrast, all the steps shown in Fig. 4 are built into a virtual appliance. To deploy the LAMP service using the

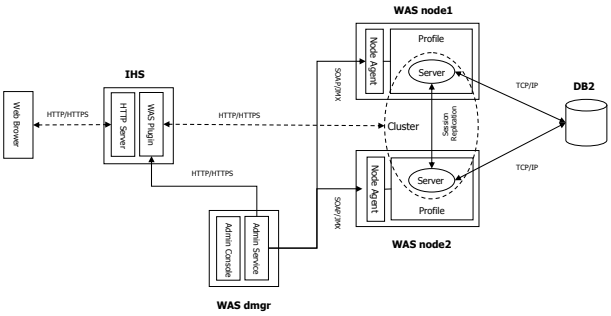


Figure 5. The topology of WAS cluster.

virtual appliance, we just need to activate the single virtual appliance image and set the appropriate network parameters.

Our second example deployment scenario is an enterprise service. In this example, we use IBM WebSphere Application Server (WAS) together with a database server and a front web server to support J2EE applications. As shown in Fig. 5, the WAS cluster consists of three nodes: one is the manager node, and the other two are managed nodes. We also apply any required patches. The database server is IBM DB2 and IBM HTTP Server is used as the web server. The deployment of the solution with traditional mechanisms is illustrated in Fig. 6.

There is fewer steps to deploy the same WAS cluster solution using virtual appliances. As shown in Fig. 7, the deployment process just sets relevant parameters among the three virtual appliances and activates them in the recommended sequence: the manager node, the first managed node, the other managed node, the web server node and the database node. Afterwards, the solution is ready for use.

5.2. Comparison Metrics

To evaluate whether virtual appliance deployment is more competitive or not, we compare virtual appliances with traditional mechanisms in terms of the deployment time, operations and parameters.

Our traditional deployment measurements are taken on the 1 Xeon 2.50GHz CPU with 1GB memory machine. Similarly, the virtual machine is a 1 CPU and 1GB memory which we deploy to a 16 Xeon 2.50GHz CPUs with 16GB memory physical machine. SUSE Linux Enterprise 10 is used for all OSs, and we use Xen 3.0.4 as the VMM.

5.2.1. Deployment Operations. Table 2 shows the comparisons of the deployment operations and time for LAMP. We first focus on the time to deploy

```
WebSphere Application Server(WAS) Cluster
1. Install WAS Network Deployment and fixpack
   in the manager node (dm);
2. Install WAS and fixpack in managed nodes (
   n1 and n2);
3. Create dmgr profile at dm;
4. Create appserver profiles at n1 and n2;
5. Start manager at dm;
6. Addnode n1 to dm;
7. Addnode n2 to dm;
8. Configure an external IBM HTTP Server;
9. Install DB2 at database node;
10. Create database;
11. Create cluster and add cluster members;
11. Configure the cluster and database;
12. Start Cluster and DB2.
```

Figure 6. Traditional WAS cluster deployment.

```
WebSphere Application Server(WAS) Cluster
1. Customize profiles in this solution
2. Activate WAS dmgr virtual appliance;
3. Activate two WAS managed node virtual
   appliances;
4. Activate IBM HTTP server virtual appliance;
5. Activate DB2 virtual appliance.
```

Figure 7. WAS virtual appliance deployment.

applications, which are 21m1s using traditional mechanisms and only 23s using a virtual appliance. Table 3 depicts the deployment operations and time for the WAS cluster enterprise solution. As shown in Table 3, virtual appliances greatly reduce the deployment time for this enterprise scenario, from 115.5 m to only 2m10s. The deployment time for virtual appliances is mainly driven by the number of virtual appliances to activate (including configuration of virtual appliances) and the time to start the software system, while the time using traditional mechanisms is dependent on the specific software system.

Please note, the total time for the virtual appliance scenario does not include the time to copy the virtual appliance images to destination directories. This time is dependent on the source media and images size. This time can be minimized by using networked storage and high speed LAN connection. In addition, the time does not include the time to install the operating system with traditional mechanisms or the time to install the hypervisor with virtual appliances. These times can be considered as constant for any applications or services, and is common skills for the deployers.

Table 2 and Table 3 also provide details on the specific deployment operations. For virtual appliances,

Table 2. Comparisons for LAMP

Traditional		Virtual appliance	
Operations	Time	Operations	Time
Install Apache	4m4s	Activate Image	20s
Install,configure MySQL	9m34s	Start servers	3s
Install PHP	7m10s	Total	23s
PHP support apache	10s		
Start servers	3s		
Total	21m1s		

Table 3. Comparisons for WAS cluster

Traditional		Virtual appliance	
Operations	Time	Operations	Time
Install was 6.0 at dm	6.5m	Activate five images	1m
Patch 6.0.2 to dm	15.5m	startManager	0.5m
Patch 6.0.2.19 to dm	9m	startNode n1	13s
Install was 6.0 at n1	5.5m	startNode n2	17s
Patch 6.0.2 to n1	13m	Start cluster	4s
Patch 6.0.2.19 to n1	9m	Start DB2,IHS	6s
Install was 6.0 at n2	7m	Total	2m10s
Patch 6.0.2 to n2	13.5m		
Patch 6.0.2.19 to n1	9m		
Create dmgr at dm	2.5m		
Create appserver at n1	3m		
Create appserver at n2	3m		
startManager	0.5m		
Addnode at n1	2m		
Addnode at n2	0.5m		
Install IHS 6.1, plugins	5m		
Install DB2 v9	6m		
Create Trade table	0.5m		
Create Trade cluster	3.5m		
Start the solution	1m		
Total	115.5m		

the deployment operations include activating and starting software system. For traditional mechanisms, the deployment operations are dependent on each software system. It is shown that for both example services, the number of deployment operations for virtual appliances is greatly reduced compared with traditional mechanisms.

5.2.2. Deployment Parameters. The large number of configuration parameters can make service deployment difficult, error-prone and time-consuming. For example, IBM DB2 has about 40 configuration parameters that are frequently customized during typical traditional deployment scenarios. In developing a virtual appliance, the builder typically exposes fewer configuration parameters than offered through traditional deployment mechanisms. Many parameters are preconfigured and not exposed during the deployment. The parameters needed to deploy LAMP from source files are: 1) Add a user and group both named `mysql` for MySQL's use; 2) Set MySQL root's password during installation; 3) Add PHP support in `http.conf`; 4) Edit the http server port for resolving possible

conflicts, for example, default server port 80 may be already taken.

Using a virtual appliance, all these parameters can be pre-configured. The only deployment time configuration is environment changes like network interface, which can be done by the appliance agent according to the activation parameters. Therefore, compared with traditional mechanisms, a virtual appliance reduces the number of configuration parameters and the difficulty of service deployment.

6. Conclusion

In this paper, we provided a model to describe the complexity of service deployment and proposed the architecture of virtual appliances. After that, we analyzed the deployment process of using virtual appliances and traditional deployment mechanisms, and compared these two deployment mechanisms in terms of the deployment time, operations and parameters. From the experimental results, we showed virtual appliances can make service deployment simpler and easier. We showed the findings apply to both a simple, open source solution (e.g. LAMP) as well as an advanced enterprise solutions like an IBM WebSphere Application Server cluster.

References

[1] V. Talwar, D. Wenchang, and Y. Jung, "Approaches for service deployment," *Internet Computing, IEEE*, vol. 9, no. 2, pp. 70–80, 2005.

[2] V. Talwar, Q. Wu, C. Pu, W. Yan, G. Jung, and D. Milojicic, "Comparison of approaches to service deployment," in *Proceedings of IEEE International Conference on Distributed Computing Systems*, 2005, pp. 543–552.

[3] A. Carzaniga, A. Fuggetta, R. Hall, A. van der Hoek, D. Heimbigner, and A. Wolf, "A characterization framework for software deployment technologies," Dept. of Computer Science, University of Colorado, Tech. Rep., April 1998, tech. Rep. CU-CS-857-98. [Online]. Available: http://serl.cs.colorado.edu/~carzanig/papers/CU-CS-857-98.pdf

[4] OMG, "Specification for deployment and configuration of component-based distributed applications," 2003. [Online]. Available: http://www.omg.org/docs/mars/03-05-08.pdf

[5] A. Dearle, "Software deployment, past, present and future," in *International Conference on Software Engineering (Future of Software Engineering)*, 2007.

[6] B. Zellen, "Easing toward utility computing," Jul. 2004. [Online]. Available: http://smbinnovator.com/index.php?articleID=3573&sectionID=4

[7] J. S. David, D. Schuff, and R. S. Louis, "Managing your total IT cost of ownership," *Communications of the ACM*, vol. 45, no. 1, January 2002.

[8] VMware, "Virtual appliance marketplace, Virtual appliances, VMware appliance." [Online]. Available: http://www.vmware.com/appliances/

[9] M. Yarnall, L. Berc, and Q. B. Wang, "Using VMware ESX server with IBM Websphere Application Server," July 2006. [Online]. Available: http://www-900.ibm.com/cn/crl/download/ESX_WAS_WP_24Jul06_final.pdf

[10] "VMware." [Online]. Available: http://www.vmware.com/

[11] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *Proceedings of ACM symposium on Operating systems principles*, 2003.

[12] "Windows installer." [Online]. Available: http://msdn2.microsoft.com/en-us/library/aa372866.aspx

[13] "Enterprise javabeans technology." [Online]. Available: http://java.sun.com/products/ejb/

[14] P. Chen and B. Noble, "When virtual is better than real," in *Proceedings of Workshop on Hot Topics in Operating Systems (HotOS)*, 2001, pp. 133–138.

[15] J. J. Wlodarz, "Virtualization: A double-edged sword," 2007. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:0705.2786

[16] R. Willenborg, "Virtual appliances – panacea or problems?" Oct 2007. [Online]. Available: http://www.ibm.com/developerworks/websphere/techjournal/0710_col_willenborg/0710_col_willenborg.html

[17] C. Sapuntzakis, D. Brumley, R. Chandra, N. Zeldovich, J. Chow, M. S. Lam, and M. Rosenblum, "Virtual appliances for deploying and maintaining software," in *Proceedings of USENIX conference on System administration (LISA)*, 2003, pp. 181–194.

[18] C. Sapuntzakis and M. S. Lam, "Virtual appliances in the collective: A road to hassle-free computing," in *Proceedings of Workshop on Hot Topics in Operating Systems*, 2003.

[19] S. Mastrianni, D. F. Bantz, K. A. Beaty, T. Chefalas, S. Jalan, G. Kar, A. Kochut, D. J. Lan, L. O'Connell, A. Sailer, G. Wang, Q. B. Wang, and D. G. Shea, "IT Autopilot: A flexible IT service management and delivery platform for small and medium business," *IBM SYSTEMS JOURNAL*, vol. 46, no. 3, pp. 609–624, 2007.

[20] VMware, "Best practices for building virtual appliances," White Paper, Nov 2007. [Online]. Available: http://www.vmware.com/files/pdf/Best_Practices_Building_Virtual_Appliances.pdf

[21] R. Willenborg, Q. Wang, D. Gilgen, and S. Smith, "Using virtual image templates to deploy Websphere Application Server," *IBM WebSphere Developer Technical Journal*, vol. 5, May 2007. [Online]. Available: http://www.ibm.com/developerworks/websphere/techjournal/0705_willenborg/0705_willenborg.html

[22] L. He, S. Smith, R. Willenborg, and Q. Wang, "Automating deployment and activation of virtual images," *IBM WebSphere Developer Technical Journal*, vol. 8, Aug. 2007. [Online]. Available: http://www.ibm.com/developerworks/websphere/techjournal/0708_he/0708_he.html