WizCloud: Simplified Enterprise Application Management in the Cloud

Su Su Xie, Rui Xiong Tian, Le He, Qing Bo Wang, Ying Chen
IBM Research - China
Beijing, China
{xiesusu, tianruix, lehe, wangqbo, yingch}@cn.ibm.com

Steve Ims, Jason McGee
IBM Software Group
Raleigh USA
{steveims, jrmcgee}@us.ibm.com

***Abstract*—Cloud computing has become a significant technology trend for the claim of business agility, scalability, reliability and pay-by-use billing model. Enterprises has been benefiting from transforming their IT infrastructure to the cloud. However, the migration of enterprise applications is hindered due to the gap between existing enterprise application model and the new paradigm in the cloud. It's still a tough decision for enterprises to take the risk of re-developing mission-critical applications, which have been proven stable for years, under new programming models and APIs, as well as the corresponding management paradigm. This paper describes an experimental application management system in Platform-as-a-Service (PaaS) named WizCloud, which intentionally targets for 1) simplifying the deployment and management of enterprise applications and 2) compatibility to conventional Java EE applications. Based on Infrastructure-as-a-Service (IaaS), WizCloud can dynamically provision a middleware environment tailored for wide varieties of enterprise applications. Current Java EE applications can be smoothly migrated to WizCloud with little investment on existing applications. Moreover, some widely used patterns of enterprise applications such as load-balancing, auto-scaling and failover have been embedded into WizCloud in form of policies, which can be customized according to Service-Level Agreements (SLAs) of applications.**

Keywords—Cloud Computing; Shared Middleware; Platform-as-a-Service;

I. INTRODUCTION

As business evolves in the past years, wide varieties of enterprises are now employing Information Technology (IT) to deliver services to their customers. Applications in enterprises are often distributed systems with ever increasing size and complexity. Investment on the development, deployment and management of those applications turns to be a significant part of operation cost. Enterprises are now pursuing cost-effective and flexible approaches on both their IT infrastructures and applications.

Cloud computing can be simply defined as a new computing paradigm where virtualized and dynamic resources are provided as on-demand services. It is attractive to enterprises due to its pay-by-use pricing model and advantages including cost saving, high availability and easy scalability. Currently, cloud computing services can be roughly classified into three major categories: 1) Infrastructure-as-a-Service (IaaS) to deliver computation, storage capabilities or networks as a service to customers. For example, Amazon provides IaaS like EC2 and S3 [1]; 2) Platform-as-a-Service (PaaS) to deliver a computing platform and solution stack as a service to customers. Google App Engine[2] and Microsoft Azure [3] fall into this category and provide the capability to implement business applications based on the interfaces exposed by such platforms; 3) Software-as-a-Service (SaaS) to deliver software to customers for use as a service on demand. Salesforce [4] and many other vendors currently provide SaaS services to customers.

Enterprises has been benefiting from IaaS, i.e. transforming their IT infrastructure based on public or private cloud service. They have seen dramatic cost saving by either outsourcing their IT infrastructure or re-organizing their internal IT resources as a centralized, on-demand service[5][6]. However, the adoption of PaaS in enterprises is hindered due to the lack of such a PaaS that is compatible with traditional middleware-based application model. PaaS in the market often provide a new set of application paradigm and runtime APIs, which imply investment on the re-development of existing applications. Moreover, enterprises often expect more control on the PaaS itself to 1) make effective use of resources according to the characteristics of their applications; 2) perform a set of conventional administrative tasks such as cross-tier optimization or data backup etc; 3) provide enterprise-grade service level assurance in terms of application performance, reliability, availability and scalability.

WizCloud arguably represents research efforts on providing a shared middleware environment where diverse existing applications can be hosted directly with little migration effort. Unlike in IaaS, where the central abstraction is *virtual machine*, the central abstraction in WizCloud is *application*. WizCloud provides necessary middleware runtimes, artifacts inventory, connectivity, and other resources to host rich set of applications. WizCloud has a viewpoint on application, supporting some particular programming models, one of which is Java EE. WizCloud also has understanding the runtime status of the application deeply. Patterns and operation experience in traditional context such like load-balancing, auto-scaling has been embedded

The paper gives an overview of WizCloud, but it does not attempt to dive into the details. The intention of this paper is to show the potential of PaaS in enterprise environments rather than a guide for practitioners to re-implement a similar system. Section 2 goes through the overall architecture of WizCloud. Section 3 describes key concepts and design principles for the understanding of WizCloud. Section 4 explains some detailed design of WizCloud. Readers can find there main problems and challenges of an enterprise PaaS and WizCloud's solution to them. Finally, we conclude our work at the end.

II. WIZCLOUD OVERVIEW

Most applications running in a enterprise are built on various middleware platforms which simplify the application development by providing a common foundation for complex low-level details such like state management, multi-threading, load-balancing, database connections, etc. The core idea of WizCloud is to provide an application hosting environment, which can reduce the cost of IT operation and application management by applying cloud computing technologies as well as deployment patterns and expertise that have been established during past years in field practices.

A. Overall Architecture

WizCloud is based on a number of design principles, which include 1) Vertical integration across the software stack; 2) Simplified deployment and management of applications; 3) Resource virtualization and sharing using cloud concept; 4) Self-management of deployments and middleware capabilities.

The high level WizCloud architecture is represented below.

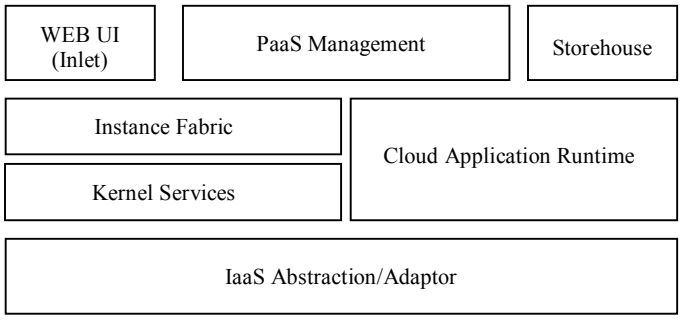

Figure 1, Overall Architecture of WizCloud

WizCloud is essentially a software management system that exposes shared middleware as a service for applications in a cloud computing environment. WizCloud allows a pluggable collection of middleware services to be exposed to applications in a way that is aligned with the application itself. Moreover, WizCloud provides a system for deploying and managing applications on a cloud with a greatly simplified and lower cost interaction.

WizCloud makes assumption of underlying IaaS to provide infrastructure resources such like virtual machines, storage and networking. By using an IaaS abstraction layer, any of those widely-accepted products in the market can be introduced into WizCloud via corresponding IaaS adaptor. Via this approach, transparency is achieved to upper layer applications, as well as the wrap of IaaS heterogeneities.

B. Core System Structure

The core of WizCloud is comprised of five fundamental sub-systems for necessary application management functionalities. We briefly go through them as below.

1) Kernel Service: The Kernel Services component (KS) is the central management agent for WizCloud. It provides the brains to manage the overall system. KS exposes a rich set of REST APIs that allow all of the management functions of WizCloud to be performed, including defining and deploying application, administering the environment, monitoring and control. KS is responsible for coordinating the interaction between applications and underlying infrastrutures.

2) Storehouse: The Storehouse (SH) is the central artifacts repository for WizCloud. All of those parts necessary to run applications live in the storehouse, including the binaries and automation scripts for products, like middleware and database software, application binaries from users, pre-populated or custom application templates etc. The storehouse also supports the association of metadata with resources and the indexing and search of contents within it.

3) Inlet: The Inlet is the user interface of WizCloud. It is a WEB-based application that provides user interfaces (UI) for interacting with WizCloud. The Inlet supports both WizCloud Users and Administrators perspectives on the system. The Inlet communicates with the Kernel Services and Storehouse components to access information about and control the system.

4) Instance Fabric: The Instance Fabric (IF) is an overly established by a set of components deployed into each virtual machine (VM). WizCloud utilizes IF as a communication basis for managing and controlling the application deployment and execution. A collection of VMs make up the deployment for an application. Each VM has an Instance Agent (IA) running. One of the IAs is elected to be the leader of the Instance Fabric for that deployment. The leader VM also contains an Instance Inlet (II) which provides a web interface for managing that deployment. In WizCloud, there will be many Instance Fabrics running simultaneously, all acting in a self-managed fashion and coordinating with the Kernel Service component.

5) PaaS Management: The PaaS Management (PM) is the sub-system for the platform administrator, which provides all necessary tools and services to manage all application deployments in the PaaS as well as the the plaform itself. Users can specify their application and corresponding deployment via various modeling tools in PM. Administrators also have their particular toolset to do system maintainence work such as monitoring, image inventory updating, application pattern creation, etc.

III. KEY CONCEPTS OF WIZCLOUD

In this section, we present a few important concepts and design details of WizCloud. They are the key to understand 1) how WizCloud significantly simplify the management of enterprise applications from a logic view, which turns the management stuffs from middleware-oriented into application-oriented; 2) how WizCloud provide the backward compatibility to conventional middleware environments, which implies existing enterprise application can run on WizCloud with little or even no development efforts.

A. *Application Model*

An application in WizCloud is tentatively organized as a combination of functionality and qualifiers relating to service-level agreements and user intention on runtime behavior of the application. The application model supported by WizCloud accordingly contains a set of application components, policies and the links between them, as illustrated as Figure 2.

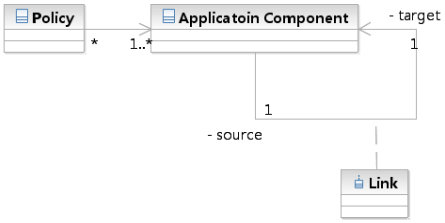

Figure 2, Conceptual Illustration of Application Model

Application components are the basic elements of an application model. They represent the middleware resources needed by the application. Each type of component contains a set of attributes for configuration either at deployment time or runtime. For example, an application server requires the configuration of the application's .war file and some version information. WizCloud provides templates of those common middleware, such as WEB server, application sever, database server, etc., to satisfy most enterprise applications. By choosing different application components as well as their attributes, users can manipulate and customize tailored middleware environment for their applications as their needs during deployment time.

A link describes one kind of dependency between the source and target application components. One type of link may also contain attributes for configuration. For example, a data source link is used to describe the dependency between the web application running on Tomcat and the database on Mysql.

Many policies could be attached to and imposed on the given application components. A policy in WizCloud contains a set of attributes that describe the expectation upon the non-functional constrains for attached components. Policies should be applicable to the attached application components.

The application model reveals the two main objective of WizCloud, i.e. the compatibility with existing enterprise applications as well as simplifying the management of them. Functionality of an application is represented by corresponding application components, which is purposely compatible with conventional models. Existing applications can thus be easily migrated to WizCloud since changes to software artifacts such like .war files and SQL scripts are not mandatory.

Policies in WizCloud are the key to simply the management of enterprise applications. In most of conventional middleware environments, there are often dedicated operation teams, rather than development teams, to handle non-functional qualifiers such as load-balancing, scalability or reliability. Expertise and deep knowledge of middleware products and solutions are required, which significantly increase the IT operation cost. WizCloud has found that these qualifiers generally are often implemented with very similar components and structures across different applications, like load-balancers, firewalls and so on. These well-understood patterns has been generalized and embedded into WizCloud. In WizCloud, an administrator no longer needs to specify an application as combination of group of middleware instances and some supporting structure like load-balancers. Instead, he should only indicate the needs of load-balancing and scalability for the application by specifying application components with appropriate policies imposed on.

B. *Software Stack As Plug-ins*

PaaS is often thought as a virtualized application hosting platform, which provides application runtime environments. Based on bare computing resources provided by underlying IaaS, PaaS need to construct software stacks for applications such as operating system and middleware. There are many options how to relate IaaS resources with software resources across applications. WizCloud adopts the most straightforward one, i.e. running middleware instances of an application on a dedicated set of virtual machines.

WizCloud is a system for enabling Platform-as-a-Service. It is composed as a base system and extensions. The base system provides a framework for user experience, i.e. UI, services to create and mange deployments of user applications and a repository for system and user artifacts. Middleware and services for user applications are extensions to the base system. Such extensions, which are called "plug-ins", provide the runtime environment for applications. Plug-ins include middleware binaries, application packages, management scripts and other software components used by application execution and management.

A plug-in is distributed and stored in WizCloud as an archive file, within which different metadata, middleware binary and all kinds of scripts are organized in directory convention. There is a compact contract of interaction between the base system and extensions. The contract defines how a plug-in is employed in application modeling and management, i.e. specification of UI generation, as well as how the base system to realize, tailor the resources in the plug-in for an application and do corresponding lifecycle management.

C. *Consistent Resource Model*

PaaS can be also looked as an integration platform for enterprise applications, where resources and services from different layers and domains converge. For example, a CRM application can be based on the enterprise's internal IaaS while storing non-critical data into Amazon S3 and invoking the Microsoft Live service for messaging. It's no doubt a huge challenge for a PaaS to consolidate heterogeneous management information from separate domains into a consistent and logical view.

WizCloud use a consistent resource model to achieve the service-management integration, flexibility and simplicity. Manageable entities are modeled as resources in granularity of management requirements from customers. These resources will be organized in tree hierarchy to represent different level of details. Each resource is defined as 1) structural information,

which is used for hierarchical resource tree construction; 2) metrics, which is the resource status to be collected and monitored at runtime; 3) properties, which is the configuration items for the resource at runtime as well as at design time; 4) actions, which is the set of operations supported by the resource.

One or more related resources are implemented as a plug-in, which specifies the codes and scripts for WizCloud to adapt to the heterogeneity of the resources. An application of WizCloud can run across multiple virtual machines, each of which contains only one part of application resources. The fully resource tree will be constructed at runtime via the agent framework of WizCloud we'll discuss later in the paper. Here we have an example how the resource tree is established for a WEB application with a two-node middleware cluster (WAS1 and WAS2) and a database server (DB2) in Figure 3 and 4.

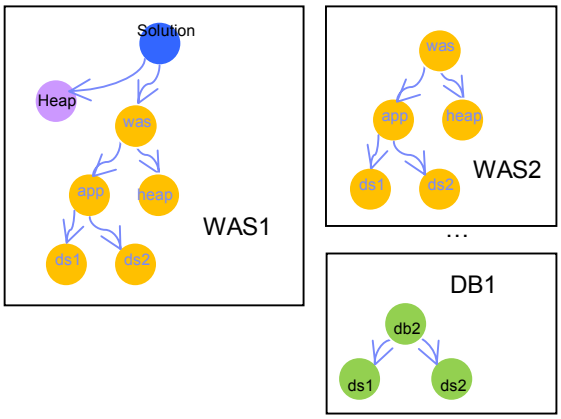

Figure 3, Resources on individual server instances

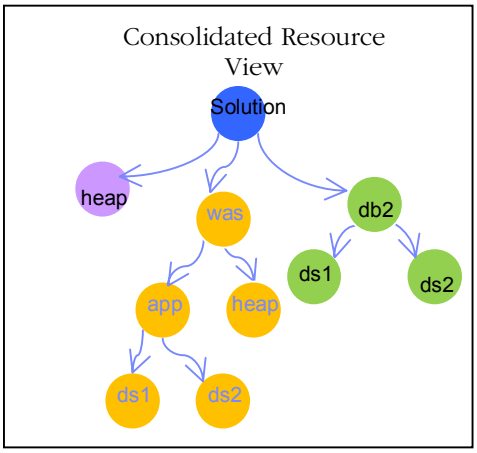

Figure 4, Consolidated Resources Tree

IV. DETAILS IN WIZCLOUD DESIGN

A. Application Instantiation

Before an application is deployed and serves their customers, PaaS needs to prepare the middleware runtime for it. As mentioned earlier in the paper, application model in WizCloud provides users only a logic view to an application. WizCloud needs to instantiate the application by provisioning customized infrastructure resources and software resources, such as virtual machines, middleware instances and specified configurations etc.

Users will specify functional requirements and non-functional qualifiers of an application as an application model. WizCloud will transform the application model into a physical layout in form of "virtual machine templates", which indicate 1) type and number of virtual machines to acquire from IaaS; 2) software stack, i.e. middleware instances and supporting software, on each virtual machine; 3) specified configurations for each virtual machines and software pieces to reflect policies and dependency between application components.

There is a clearly defined interaction interface between the base system of WizCloud and plug-ins for the transform. On the user's request for application deployment, WizCloud performs the transform process automatically by invocation on plug-ins specified in the application model. Since no human intervention is required, it significantly reduces the effort for application deployment, as well as the expertise of the user. It's up to the plug-in developer to decide how to tailor the software resources, i.e. middleware and their configurations, according to related stuffs in the application model.

B. Agent Framework and Application Activation

There is an agent framework in WizCloud that runs across all virtual machines of an application. Each agent is composed of three components as follows:

1) *Agent runtime*, which serves as the runtime environment to handle RESTful requests, as well as transmission protocol related issues. On arrival of requests, it maps the requests to corresponding service interfaces and invokes the related services.
2) *Service interfaces*, which are interfaces to be invoked by agent runtime to perform some functions. Some of the interfaces are implemented by WizCloud itself to provide a set of common services across enterprises or applications. These interfaces and their corresponding implementations are built-in services. Other interfaces are left to plug-in developers for middleware-specific or system-specific functions. These interfaces and their implementation are for extended services.
3) *Metadata and scripts*, which are the incarnations of the service interfaces. Each service may depend on some metadata and is executed through one or more scripts. Shell scripts and Python scripts are intrinsic to be supported by WizCloud. However, other kinds of scripts can also be supported so long as corresponding agent runtime are loaded.

In current WizCloud implementation, the agent runtime is based on Simple Agent Framework (SAF) which includes an execution engine, metadata handler, agent runtime, and REST framework. It is responsible for receiving RESTful requests and invoking relevant services to support the management functions. Services are implemented for corresponding management functionalities at different levels. For example, to support the starting/stopping of an application, application availability service and appliance availability service should

be in place. They are responsible respectively for coordination of the actions taken for all related components in the application, and, for acquiring or releasing all the bare computation resources such as virtual machines.

C. *Application Tweak at Runtime*

In conventional middleware environment, the structure of an application, which includes hardware and software stacks, configuration for the middleware and the number of instances, is relatively static. However, PaaS has the potential of re-shaping the applications structure dynamically since resources can be provisioned at runtime. Due to auto-scaling and high availability policies, virtual machines as well as middleware instances can be created or destructed automatically from time to time according to runtime status of an application.

In PaaS, users have less control of the platform as before. For example, the exact middleware instances existing at any given point of time are determined automatically according to pre-set policies. This invisibility to the runtime structure of application raises an important question of how to do (re-)configuration on a running application with diverse concerns due to reasons such like external requirement change or optimization. Traditional management tools will fail here because of the gap between the logical view of the application and its physical structure. The problem gets even harder because direct touch on middleware instances is often required in many of enterprise environments. Conventional administrative tasks and skills, such as cross-tier optimization, still make much sense there.

WizCloud provides the capabilities of application tweak, i.e. changing an application's properties at runtime, at two levels. At the first level, users can do application-centric tweak in a view aligned with the application model. It implies that users can stay at a higher level and speak in the language of their intentions instead of concrete configuration items of related middleware. The second level provides a resource-centric view. Users can change middleware's behavior so long as plug-in developers provide the support in the resource model.

WizCloud provides necessary consolidation at both levels, i.e. applying configuration change to all related middleware instances regardless the dynamics of application structure. WizCloud will maintain the up-to-date configuration of the application and guarantee eventual consistence of middleware instances. For example, if one virtual machine happens to crash before the request's arrival, the new set of configuration items will be applied when the virtual machine is restarted. Moreover, to keep application continuity, WizCloud adopts a rolling strategy during application tweak if the configuration changes require restart of the middleware or virtual machines.

D. *Flexible Policy Support*

On of promising feature of cloud computing is its pay-by use model, which implies more effective resource use and less human intervention for the application in PaaS. To achieve that, PaaS have to adopt a policy-based management approach to dynamically change the resource set and behavior of applications. In another viewpoint, policy objectives are particularly difficult to meet in the multi-application setting of PaaS due to potential resource contention.

Given the feature of fast and dynamic resource provision, WizCloud is trying to provide a policy framework that is more flexible than in conventional middleware environments. Based on the consistent resource tree discussed in the earlier section, WizCloud can support policies on finer grained resources across the boundary of middleware instances. For example, Users can be charged on the database connection pool size. It's very natural to specify a policy to get the pool size increase in peak hours and shrinks on light workload.

There is an experimental policy sub-system in WizCloud to support that type of policies. Users can express their non-functional qualifier in a comprehensive form. For example, 1 million concurrent users with response time less than 1 second; elastic database connection to guarantee the request rate of no less than 1000 per second. The implementation of policies will be packaged as plug-ins of WizCloud. Policy developers will transform the application level policy specification into some execution over the application's consistent resource tree. Such execution often includes metric collection, condition check and action invocation of related resources, which is in form of some policy compliance scripts in a particular domain language. At runtime, there is a policy engine in WizCloud that periodically collect the metrics of all resources in the application and check against the condition. Whenever some condition is triggered, the policy engine will schedule a series tasks to change the behavior of affected middleware or invoke kernel service to do resource related work.

CONCLUDSION

In this paper, we present the key ideas on the design and implement of an enterprise-oriented PaaS, which simply the deployment and management of enterprise applications. And more importantly, the platform is compatible with conventional middleware models, which implies the avoidance of re-investment on acquiring or development for those existing applications. The future work of WizCloud involve two aspects: 1) design and implement the management agent framework to make it general to diverse applications; 2) explore computing resource optimization vertically, not individual PaaS layer to meet economic and application performance requirement.

REFERENCES

[1] Amazon Web Services, http://aws.amazon.com.

[2] Google App Engine, http://code.google.com/appengine/.

[3] Microsoft Windows Azure, http://www.microsoft.com/windowsazure/windowsazure/.

[4] Salesforce, http://www.salesforce.com.

[5] Xing Jin, Ruth Willenborg, et al, "Reinventing virtual appliance", IBM Journal of Research and Development, 2009.

[6] Xinhui Li, Ying Li, et al, "The method and tool of cost analysis for cloud computing", IEEE international conference on cloud computing, 21-25 Sep. 2009.