ISSN: 2277-9655
Impact Factor: 4.116

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

A NOVEL TECHNIQUE FOR SCALABLE, EFFICIENT DEPLOYMENT OF SOFTWARE IN CLOUD COMPUTING ENVIRONMENTS

S.Venkataramana*, Dr. P.V.G.D.Prasad Reddy, Dr S.KrishnaRao
*Research Scholar, Dept. of Computer Science & Systems Engineering,
Andhra University, Visakhapatnam, Andhra Pradesh, India
Dept. of CS&SE, Andhra University, Visakhapatnam, Andhra Pradesh, India
Dept. of IT, Sir CRR Engineering College, Eluru, Andhra Pradesh, India

DOI: 10.5281/zenodo.55615

ABSTRACT

In modern computing environments (Clouds, Data Centers, multi-tiered applications, etc.) deployment of software encounters the obstacle of handling heterogeneity. Real-world solutions are built by combining various technologies, programming languages, platforms, Operating Systems, and applications. The deployment mechanism not only has to adapt to a changing environment but it has to also respond gracefully to unanticipated demand surges by scaling-out and scaling-in to meet performance requirements. For these reasons, deployment of software in heterogeneous environment is a challenging task. Efficient resource provisioning plays a key role in ensuring that global service providers (like Google) adequately accomplish their performance obligations to customers while maximizing the utilization of the underlying environment. In this paper, we propose a novel deployment mechanism which can efficiently provision, de-provision and maintain services in heterogeneous environments. Our approach creates software packages by embedding parameter values within binaries in a way which is compatible with most target environments allowing for seamless provisioning, tracking, maintenance and de-provisioning of services in a robust manner without requiring specialized agents deployed in target environments. We have validated our mechanism by constructing prototypes and the results demonstrated that it is possible to scale-out and scale-in rapidly in response to the demand fluctuations by automating deployment mechanism in a heterogeneous environment.

KEYWORDS: Radia, Heterogeneous Environments, seamless provisioning, JuJu, OpenStack, SmartFrog, Aggregation, Generic Provisioning Function, AggMech.

INTRODUCTION

Cloud computing has grown up increasingly pervasive, providing end-users with temporary access to scalable computational resources. At conceptual level, cloud computing is a good fit for technical-computing users. The scientists are ready to take advantage of cloud computing resources to execute scientific workflows [1], [2], [3]. However, the current cloud computing market developed gradually by the Software Industry, it is not a good match for the needs of technical-computing end-users from the especially high-performance computing (HPC) community. Providers such as Rackspace & Amazon provide users with access to a homogeneous set of commodity hardware; in details of the hardware obscured through virtualization technology and little or no control of locality (accept sometimes by geographic region). By contrary to, technical-computing end-users want to obtain access a heterogeneous set of resources, such as different network interconnects, accelerators and Machine Architectures.

WHY A HETEROGENEOUS ENVIRONMENT IN CLOUD?

Data centers and Computational centers are usually limited by power density and efficiency, and compute density. General-purpose microprocessors and server manufacturers are working to improve efficiency of power; heterogeneous processing resources should provide an order of magnitude or more improvement using these metrics.

The improvements are seems to be persistent, because specialized devices can be optimized for specific kinds of computations, and for efficiency. There are many examples for problems well suited to specific architectures. Examples of such architectures include network packet processors, graphical processing units, digital signal processors, symmetrical multiprocessors (SMPs), and conventional CPUs. Today's cloud Environment, with a few notable exceptions (e.g. R Systems, SGI Cyclone, Amazon Cluster GPUs), generally focuses on hardware commodity, with no control over target architectures and a fixed number of memory/CPU sizes. If cloud users are able to take the advantage of the performance and efficiency of heterogeneous computing, the cloud infrastructure environment software must recognize and handle this heterogeneity. In the past, grid computing and batch scheduling have been commonly used for large scale computation. Cloud computing presents a different resource allocation paradigm than either grids or batch schedulers. In particular, Amazon EC2 [5] is equipped to handle many smaller compute resource allocations, rather than a few, large requests as is normally the case with grid computing. The introduction of heterogeneity allows clouds to be competitive with traditional distributed computing systems, which often consist of various types of architectures as well. When combined with economies of scale, dynamic provisioning and comparatively lower capital expenditures, the benefits of heterogeneous clouds are numerous. Cloud computing allows individual users to have administrative access to a dedicated virtual machine instance. The capability to separate users is superior compared to a batch 2011 IEEE International Conference on Cluster Computing 978-0-7695-4516-5/11 $26.00 © 2011 IEEE DOI 10.1109/CLUSTER.2011.49 378 scheduling approach[8], where it is common for multiple jobs to share a single operating system. The advantages of this are apparent from the perspectives of security as well as flexibility for users, offering a variety of operating systems.

In a large-scale datacenter environment, different workloads are paralleling serving the end-user needs. Most of the workloads deployed as per compliance check, supportability on different infrastructure to serve the end-user needs. Managing the workloads by the administrator for supportability and maintainability is a herculean task, since the administrator need to understand entire heterogeneous environment [9]. Most of the customers go with heterogeneous datacenters to overcome vendor locking and at the same time to ensure workloads seamlessly providing required services to the end-users. For heterogeneous resources it will allow for deploy various workloads on different environments to avoid single point of failure and reliability for the resources.

PRIOR SOLUTIONS

Multiple organizations are developing and prototyping the client automation tools. Among these tools, few of them are popular based on customers base, below are the high level details below:

JuJu enables provisioning of entire environments in the cloud with few commands on public clouds like popular public cloud services Amazon, Rackspace, to private clouds built on OpenStack, or raw bare metal via Messaging as a Service [4. The enabling components are its models called charms that are developed for multiple target computing environments and subcomponents.

Radia is a client server solution to provision and maintain software on a limited set of popular OS platforms that can be provisioned using installation wizard, logon script, and management portal. Radia Server stores data for all applications including desired state, and maintains the policy. Basic functions (Policy Resolution, Desired State Resolution, and Data Download) are performed by Radia client which is target platform specific. Radia utilizes multiple level caches for optimization.

SmartFrog is Java-based software framework for configuring, deploying and managing distributed software systems that might contain multiple components that need to interoperate. SmartFrog consists of a language for defining configurations, providing powerful system modeling capabilities and an expressive notation for describing system configurations, a secure, distributed runtime system for deploying software components and managing running software systems, a library of SmartFrog components that implement the SmartFrog component model and provide a wide range of services and functionality.

Chef is a systems and cloud infrastructure automation framework to deploy servers and applications to any physical, virtual, or cloud locations. The chef-client relies on abstract definitions (known as cookbooks and recipes) that are written in Ruby and are managed like source code. Chef uses chef-client to execute deployments independently.

PROPOSED SOLUTION

The above mentioned tools try to resolve the deployment of software across heterogeneous hardware. The proposed solution solve the below mentioned problems:

1. The proposed solution resolves software deployment mechanism for heterogeneous environments which restricts heterogeneity to software packaging while retaining deployment requests and deployment engines abstract (heterogeneity agnostic) and automates the software packaging deployment.

2. Further the proposed solution optimizes the mechanism of the environments with specific constraints and physical hardware and slow network.

The proposed solution packages software components in a unique way by first creating a common binary for the varied environments and then uses a context sensitive installation script to replace the variants for each different environment with specific values specific to the target environment. This installation script is capable of retrieving parameters from varied locations within an installed environment. A regressive method to eliminate the differences between the diverse environments as explained in steps a, b, and c below helps to make the provisioning mechanism fully-automatic [12].
Existing provisioning tools could be described by the following function:

Tsconf = Dr (SM1, SM2, … SMn)

Input to provisioning function package in the form of consumable by provisioning system results in fully functional software component installed on target system. Software component would typically consist of installation binary and installation /configuration instruction to configure software.

METHODOLOGY

The proposed methodology tries to automate the software deployment in a mathematical approach.
The function defined above considered as

Tsconf = Target System Configuration;
Dr = Deployment Routine
SMn = Software Model Components.

Below is the mathematical function defined for automating software.

1. Initialize the Tsconf and a temporary Tempsconf variables.
2. Find the cumulative of base software model component and cumulative of Layered Architecture and configuration update model data.
3. Compute the cumulative of the I..n software model components and assign to Tsconf

Tsconf = Dr (Cumulative (Smbase(SM1)) + Cm (SChierarichal (SM1) + Cumulative (SC)+ …
+Depenv(CfgUpd))

Tsconf	Target System Configuration
Dr	Deployment Routine
Cm	Cumulative mechanism for cumulative generic interface calls
Depenv	Dependency environment
Smbase	Base Software Component
SChierarichal	Hierarchal Software Component
SCi	Software Component
CfgUpd	Configuration update

Unified mechanism to make generic interface calls to enable aggregation of various heterogeneous software components which includes software libraries and configuration script. Generic interface is possible through elimination of parameters from provisioning function call as some of potential parameters would be encapsulated within binary components, common environmental parameters like hostname and IP would be stored in predefined standard location, and parameters that configure integration between components when not encapsulated into binary would use local relative references whenever possible.

Designer can create or generate a trivial provisioning function Dr (installation script with single parameter which uniquely identifies application like a customer name) while all other localization parameters would be found in a standard location on the target system defined by Cm function (embodiment could be a property text file). The deployment function would operate on the base Software Component i.e., Smbase, which is derived from manually installed software component SCi files to specific location with configuration files embedded or replaced with stubs. The deployment function could also operate on hierarchal Software Component i.e., SChierarichal, which is comprised of multiple software components pre-combined into single Software package, so there is no need to pass parameters that integrate them. One of the Software components could be a base component which would place Cm parameters into standard location, for installing OS and save parameters in the standard location. The deployment function could also operate contains environment configuration changes Depenv is a representative for OS .reg file that would self-extract configuration parameters to Windows registry to complete deployment of target operating environment.

EXPERIMENTAL RESULTS

Overall, proposed method allows constructing a complete target environment of any complexity by utilizing generic provisioning function without need to pass environment parameters in traditional approach. This enables to configure software components between heterogeneous systems in a cloud environment without pre-validation of communication specifications between the nodes.

For creating a single installation routine of heterogeneous environments which are capable of installing the entire provisioning stack by partitioning each software component of the stack into binary and a context-sensitive installation script that is capable of extracting Cm parameters for location within an install environment.

The deployment method is in a combination of manual deployment method and model-based provisioning method. Manual deployment is used in design phase to create Software binary subcomponent. In runtime phase proposed used model is based on provisioning to deploy various software components to heterogeneous target environments. This model is a minimalistic model predicated on intrinsic native properties of software components: installability, executability and localizability.

Standardizing on our deployment method that would enable context sensitive patching without requiring a deployment agent installed on a managed device. A generic scheduled upgrade script could pull binaries of newer versions of software components installed while preserving local configuration parameters. Agentless provisioning would not require remotely accessible identification and agent maintenance.

ISSN: 2277-9655
Impact Factor: 4.116

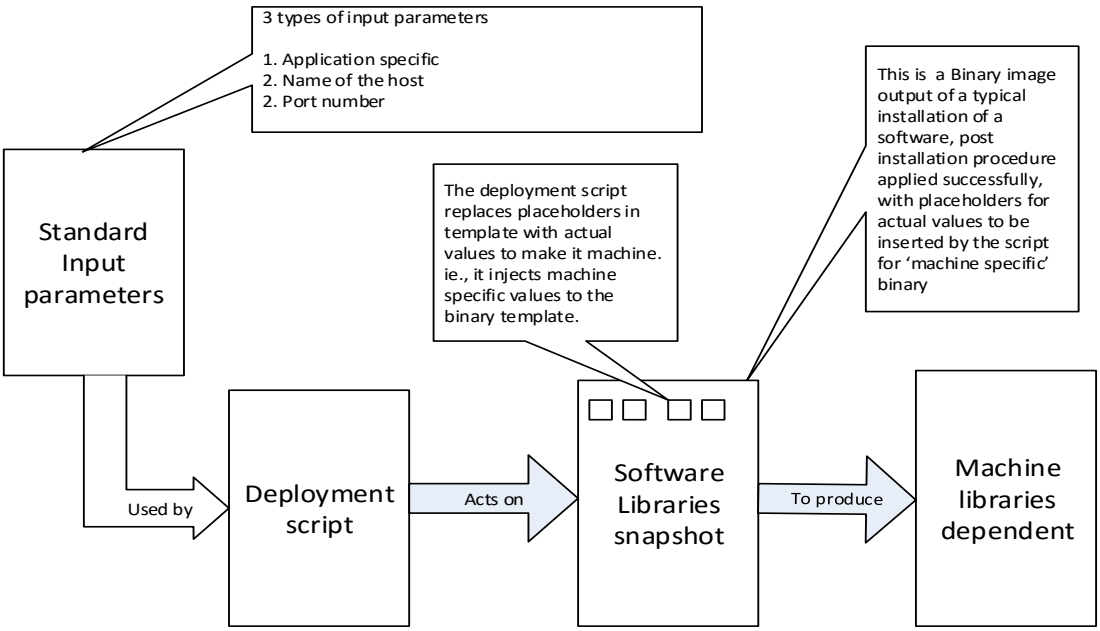

Figure-1: Proposed methodology for software deployment

AUTOMATING SOFTWARE PACKAGING

Cumulative mechanism for cumulative generic interface calls, Cm is a method to automate model-based provisioning by leveraging unified model inputs required for software installation and requirement to package software component in a specific way. This is possible because of essential properties of commercial software, that is, it is installable; its installation results in creating files in certain locations on the target system; there are limited number of standard configurations deployed in specific enterprise environment; an installation input parameter or its derivative is stored within software image installed on target system in specific location. This enables the following build engineering process to make repeatable:

1) Identify standard target environment (e.g. Operating system, file system layout and target location for software)
2) Identify standard configuration (e.g.: installation option selections, standard binary location and tuning parameters)
3) Identify naming to identify instance (e.g. unique application identifier or application short name)
4) Identify installation target (e.g. hostname which could be derivative of short name, FQDN)
5) Identify locator function of target device (IP, network interface name, default gateway)
6) Identify names of personal to be supporting software component(to generate OS user IDs and groups)
7) Identify addressable identification of other software components to interoperate with (remote hostname or IP).

Installation instances have the same target binary with exception of these configuration parameters or its derivatives (for example, encryption keys/certificates). Thus, using standard installation process, since each software product is installable, an engineer could produce binary package for software provided configuration and target environment are standardized. Then simple script could be developed to replace configuration values within binary based on this standard set of input parameters.

ISSN: 2277-9655
Impact Factor: 4.116

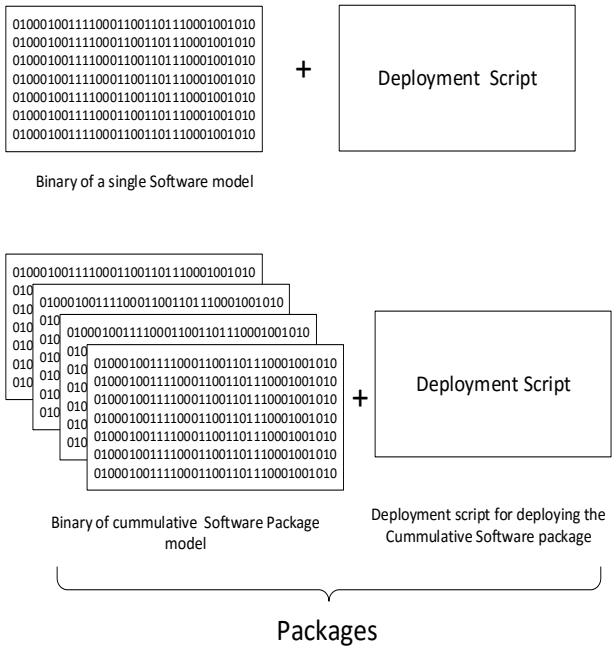

Figure 2: The resultant components of the deployment mechanism

CONCLUSION

Frequently deployment of software reaches beyond one or more software components onto provisioning of a complete functional compute. It is possible to deploy a complete compute including all OS configurations, software components and applications via cumulative as described above. However, this cumulative might not be practical to accomplish due to network limitations when a prolonged bi-directional communication is not feasible, due to variability in physical hardware platforms requiring specialized software components (drivers), or due to frequent changes in target operating environments.

The network limitation is overcome by cumulative all components into a single self-installable media that is installed or mounted in the way that is not affected by network limitation. Creation of this media is automated by the design described in the second embodiment.

The variability limitation is overcome by cumulative bundle via dynamic context-sensitive replacement, removal or injection of software components containing required drivers. These software components are packaged via the design technique described above. Changes in target operating environment are overcome by dynamic context-sensitive replacement, removal or injection of software components required for new target environment and by replacing binary portion of software components with versions compatible with new target environment. When source code is available binaries are dynamically compiled into version compatible with new target environments just in time for provisioning.

REFERENCES

[1] K. Keahey, “Cloud computing for science,” in Proceedings of the 21st International Conference on Scientific and Statistical Database Management, New Orleans, LA, June 2009.
[2] E. Deelman, G. Singh, M. Livny, J. B. Berriman, and J. Good, “The cost of doing science on the cloud: the montage example,” in Supercomputing Conference, 2008.
[3] L. Wang, J. Tao, M. Kunze, A. Castellanos, D. Kramer, and W. Karl, “Scientific cloud computing: Early definition and experience,” in High Performance Computing and Communications, 2008. HPCC ’08. 10th IEEE International Conference on, sept. 2008, pp. 825 –830.
[4] “OpenStack.” [Online]. Available: http://www.openstack.org.
[5] “Amazon elastic compute cloud (amazon EC2).” [Online]. Available: http://aws.amazon.com/ec2/

[6] Z Sanaie, s Abolfazli, A Gani., Heterogeneity in mobile cloud computing: taxonomy and open challenges-Surveys & Tutorials,2014- ieeeexplore.ieee.org.

[7] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A Berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, Feb 2009. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS- 2009-28.html

[8] Steve Crago, Kyle Dunn, Patrick Eads, Lorin Hochsteiny, Dong-In Kang, Mikyung Kang, Devendra Modium, Karandeep Singh, Jinwoo Suh, John Paul Walters., " Heterogeneous Cloud Computing" Conference: 2011 IEEE International Conference on Cluster Computing (CLUSTER), Austin, TX, USA, September 26-30, 2011

[9] BP Rimal, Echoi, I Lumb., A Taxonomy and survey of cloud computing systems., INC, IMS and IDC, 2009. NCM'09- ieeeexplore.ieee.org.

[10] A Klen, C Manweiler, J Schneider., Access Schemes for mobuile cloud computing, 2010 Eleventh...,2010-ieeeexplore.ieee.org.

[11] VC Emeakaroha, I Brandic, M Maurer., SKA-Aware application deployment and resource allocation in clouds., Computer Software...,2011- ieeeexplore.ieee.org.

[12] S Yeo, HHS Lee., Using mathematical modeling in provisioning a heterogeneous cloud computing environment., computer,2011-ieeeexplore.ieee.org, Vol:44, Issue: 8, 2011, pp.55-62,ISSN:0018-9162,DOI:10.1109/MC.2011.96.

[13] H Takabi, JBD Joshi, GJ Ahn., Security and Privacy challenges in cloud computing environments., IEEE Security & Privacy, 2010-computer.org.