International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 11, November - 2013

Auto-Deployment of Multi-Tier System in Hybrid Cloud Environment

[1]Kamalesh Karmakar, [2]Priya Roy
[1]*Department of CSE & IT, Meghnad Saha Institute of Technology, Kolkata, India.*
[2]*CSE, Jadavpur University, Kolkata, India.*

Abstract

A large complex Multi-Tier system can be deployed in Hybrid Cloud Environment according to user's need. Present paper discusses Multi-Tier System Deployment process in Hybrid Cloud Environment using Workflow along with an example. The definition of Multi-Tier structure itself dictates the deployment process, leaving the scope of manual intervention. The system uses the facilities of scaling up & down to utilize the resources optimally in case of dynamic user requests.

Keywords *– Hybrid Cloud, Multi-Tier System, Workflow, Workflow Management System.*

1. Introduction

A Workflow consists of a sequence of connected operations declared as work of a person, a group of persons, an organization of staff, or one or more simple or complex structure. Workflow serves virtual representation of actual work and it may be seen as an abstraction of real work segregated in work-share, work split or any other types of ordering. The described flow often refers to a document, which is being transferred from one step to another. A workflow is a pattern of activity enabled by a systematic organization of resources, defined roles and information flows, into a work process that can be documented. Workflows are designed to achieve processing intents of some sort, such as physical transformation, service provision, or information processing. The term 'workflow' is used in computer programming to capture and develop human-to-machine interaction.

Workflow Management System is developed to automate and control business process [1, 2]. According to user's requirement, data store and web services cross organizational boundaries [3, 4]. A substantial work has been done for workload distribution in heterogeneous computing systems in efficient ways and has been published in [5]. Workflow based process control has been defined along with design and application in [6].

In present scenario Virtual Machines (VMs) are deployed in Hybrid Cloud Environment [7] according to user's request. User should run VMs first and then configure them to build up a computational Cluster or a Multi-Tier System according to their need. This manual process is very hard task to a System Administrator. To reduce this effort of an Administrator and to optimize start up & configuration time a Management System is being proposed for Hybrid Cloud Environment. Here, before VMs deployment, a Deployment Descriptor will be generated mentioning required software installation & configuration; and VMs will be deployed accordingly.

A Workflow Management System for Hybrid Cloud Environment (WMSHCE) presented here manages and defines a series of computational tasks within an organization to produce a final outcome(s). WMSHCE allows users to define different workflow for different types of jobs or processes. At each stage in the workflow, one individual or group is responsible for a specific task. Once the task is completed, the WMSHCE ensures that the responsibilities for the next task for an individual get notified. It also ensures that the data required for execution are also received. The designed WMSHCE can optimize redundant tasks and ensures incomplete tasks to be completed.

In this context in the following sections WMSHCE will be described in details. In section-2 Multi-Tier System has been defined followed by working principle of Workflow in Cloud Environment in section-3. In section-4, a Workflow generation technique has been defined for Hybrid Cloud Environment, followed by Multi-Tier System Deployment and Monitoring in HCE using Workflow.

2. Multi-Tier System Definition

On this background these subsections discuss the definition of Multi-Tier System in general and its applicability in HCE.

2.1. Multi-Tier System --

In traditional system users can run different application(s) in different tier(s), i.e. Web-Tier, Application-Tier and DB-Tier. User can deploy a system with one or multiple tiers containing one or more servers in every tier performing specific responsibilities.

2.2. Multi-Tier System in HCE—

The main reason behind Multi-Tier System deployment in HCE is to distribute the workload among different tiers with varying number of servers to provide scalability. Once a system gets deployed, it is always possible to add/delete tier(s) itself or to add/delete servers in a specific tier.

The relationship among multiple servers is defined in the Multi-Tier System-Definition file. Every tier has a unique name among all tiers in this descriptor. This helps users locating servers contained in this tier later on in actions. The main responsibility of the tier startup or shutdown sequences are to startup or shutdown the servers contained in it. Servers are the atomic unit of artifacts in the Multi-Tier System Definition and represent the actual machine instances to comprise the Multi-Tier system.

On this background the following section discusses the working principle of generic Workflow and a demonstration of a Workflow in cloud environment.

3. Working Principle of Workflow in Cloud

Workflow automates the transfer of information to support the flow of work of business process governed by rules or procedures to deploy any system or service automatically, therefore delays are minimized, processes get speed up, and cost savings are realized. A workflow consists of a sequence of connected steps defined in a XML file, known as System Definition File (SDF). A sample SDF file structure has been shown in appendix. In case of parallel processing where two or more tasks are performed concurrently, workflow is important to resolve their dependencies. Therefore the workflow is very important for the computing in Cloud and for the end users. WMS itself eliminates unnecessary steps to improve efficiency and controls whole process based on user's requirement. Workflow empowers administrators to manage the business process under the control of WMS. The procedures are formally documented and followed exactly, ensuring that the work is performed automatically in the way planned in SDF. The term flow refers to two things, one is referring the flow of information from process to process and the other refers the flow of activities to be done.

The deployed systems are generic and can perform synchronous or asynchronous tasks. If at that moment system deployment is not possible in Hybrid Cloud Environment, the workflow does not block any running service, it just informs the user replying with a pending message. It controls the delivery of process of applications or services in on-demand Cloud Environment. Here OpenSymphony [8] Workflow has been used to deploy the systems on the Cloud Environment.

In existing system, VMs are deployed in Cloud Environment manually. In this proposed model VMs are deployed in Cloud Environment using Workflow to reduce deployment and configuration cost. In the remaining section the background of designing a Workflow for Cloud Environment will be discussed.

Here a scenario is being considered to deploy Multi-Tier System. A computational Cluster consists of multiple servers. Here one server acts as front end to distribute workload among other Computational Nodes (CNs). Computational nodes are registered to Cluster Controller (CC). To deploy this Cluster, firstly CC should run before deploying CNs. When CC gets an IP address CN will be deployed. Post-script will run in CC to register CNs to form a Computational Cluster. Based on resource usage Computational Cluster can be scaled up/down.

The SDF file is generated after submitting user's requirement from the User interface (shown in Figure 5).

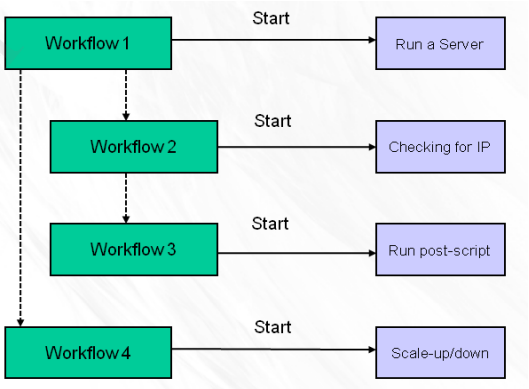

Figure 1. Working Principle of Workflow

In Figure-1 working Principle of Workflow has been defined with respect to Cloud Environment. In the proposed model, firstly Servers are booted and start execution based on Workflow1 defined in System Definition File (SDF). Later on program control goes to Workflow2 to check IP addresses of those deployed Servers. After getting IP addresses, program control is transferred to Workflow3 to run post-script commands. In Workflow4, based on resource utilization Systems are scaled-up or scaled-down to provide high performance in optimal way considering 80% of resource utilization as critical value.

When new job request comes to the system and resource utilization is more than the critical value, the system needs to scale-up for assigning the task to the new server. When resources are underutilized new jobs

are distributed in such a way that few servers can be shut down. During scaling up the number of server added to the system cannot exceed the maximum value of required server as defined in SDF. The similar concept is applicable during scaling down, where each tier should have at least one server. If the minimum and maximum values of required servers are same then the system could not be scaled up or scaled down. In this situation if new job request arrives and resource utilization is more than the critical value then the job could not be assigned to any server immediately, those will be queued.

In this context following section describes the working principle of OSWorkflow, design technique of proposed Workflow for Cloud using OSWorkflow.

4. Designing Workflow for Cloud using OpenSymphony for Proposed Model

OpenSymphony is an Open Source project dedicated to providing enterprise class J2EE components. OSWorkflow is fairly different from most other workflow systems available, both commercially and in the open source world as it's extremely flexible can be plugged in to almost any need or existing application. OSWorkflow can be considered as a low level workflow implementation. Situations like loops and conditions that might be represented by a graphical icon in other workflow systems must be coded in OSWorkflow.

OSWorkflow is based on the concept of the Finite State Machine allowing a simple XML file to be translated into business workflow process. Each state is represented by the combination of step ID and status. A transition from one state to another occurs, when an action is performed. There is always at least one active state during the lifetime of a workflow.

Here is a code segment to define the 'state' of a machine-

<unconditional-result old-status="Finished" status="Queued" step="2" />

The following line shows the way how an action is performed in a Workflow-

workflow.doAction (workflowId, stepId, information);

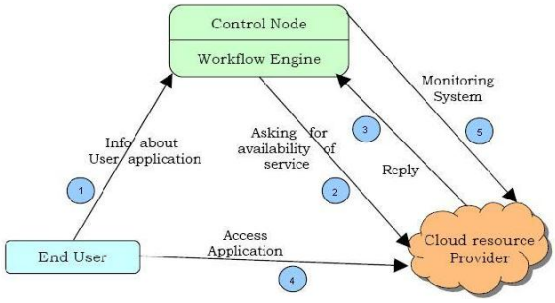

Figure 2. Architecture of Workflow in Cloud

In this context the Architecture of Workflow is being described as shown in Figure 2. The Workflow engine is a component in a workflow automation program that knows all the procedures, steps in a procedure and rules of each step. The workflow engine determines whether a process is ready to move to the next step. Here the Workflow Engine controls all the steps and actions performed by the Workflow. Control node collects the information about user application from the end user over Internet. Then it checks for the availability of services from the cloud resource provider. If the service is available but not running in the server then the workflow first starts up the service and then give permission to end user to access the application. If there is no service available right at that moment, it informs the end user replying with a pending message for some time. After giving the permission to access the application control node monitors the whole automation process.

4.1. Steps & Actions

A Step is simply a workflow position. As a simple workflow progress it moves from one Step to another. Actions specify the transaction that can take place within a Step. At the beginning of Workflow Steps are not defined. The user must take some Actions to start up the processes and to set possible Actions to start the Workflow specified in <initial-actions>.

In Figure-3 an initial action is defined for starting a Workflow and after finishing this action the flow of control will go to step-1.

```
<initial-actions>
   <action id="1" name="Start Workflow">
      <results>
         <unconditional-result old-status="Finished"
status="Queued" step="1"/>
      </results>
   </action>
</initial-actions>
```

Figure 3. Structure of Initial Actions

Two actions (ex. runInstance and copyFile) have been shown in the following Figure 4. The old status attribute is used to denote what should be entered in the history table for the current state to denote that it has been completed. In almost all cases, this will be 'Finished'.

```
<steps>
<step id="1" name="Starting Step1">
<actions>
<action id="2" name="runInstance">1.12"
<results>
<unconditional-result old-status="Finished"
status="Underway" step="1"/>
</results>
<post-functions>
<function type="beanshell" name="bsh.function">
<arg name="script">
<![CDATA[
import com.xerox.amazonws.ec2.Jec2;
import com.xerox.amazonws.ec2.LaunchConfiguration;
import com.xerox.amazonws.ec2.ReservationDescription;
import com.xerox.amazonws.ec2.InstanceType;
ReservationDescription runInst =
transientVars.get("ec2").runInstances(transientVars.get("lc
"));
]]>
</arg>
</function>
</post-functions>
</action>          <action id="3" name="copyFile">
<results>
<unconditional-result old-status="Finished"
status="Queued" step="2"/>
</results>
<post-functions>
<function type="beanshell" name="bsh.function">
<arg name="script">
<![CDATA[
.....
]]>
</arg>
</function>
</post-functions>
</action>
</actions>
</step>
.....
<steps>
```

Figure 4. Structure of Steps and Actions

The actions as specified above are of limited use. For example, it is possible for a user to call action3 without first having called action2.

4.2. Workflow Status

It is a string, and it describes the status of a workflow within a particular step. A workflow consists of multiple steps to represent the flow. For the current step, there may be multiple actions. An action may be set to run automatically or be selected to be run programmatically through user interaction. Each action has at least one unconditional result and zero or more conditional results. If multiple conditional results are specified, the first result for which all conditions are met is executed. If no conditional results are specified, or if no conditions are satisfied, then the unconditional result is executed.

In this context the next section discusses Multi-Tier System Deployment & Monitoring in Cloud Environment.

5. Multi-Tier System Deployment & Monitoring Using Workflow in Cloud

In this proposed model to deploy a Multi-Tier system in Hybrid Cloud Environment a SDF is required. This file consists of all the information about the system that contains how many tiers and how many servers to launch, which tier consists how many servers and the specification about the servers. To deploy a system the user just need to click the Deploy button after providing information about the system through this Hybrid Cloud web interface. User does not need to write any SDF. The system definition file will be generated automatically getting input from the user. A system can contain one or more than one tier and a tier may contain one or more than one server. Thus a user can get a Multi-Tier system containing many tiers and servers.

A web interface (shown in Figure 5) has been developed to deploy Multi-Tier system in Hybrid Cloud Environment. The user can set some 'post-startup' functions before system deployment. This 'post-startup' command will be executed just after running a server.

The designing of a system-definition file can be divided into two steps - the first step is to define the system artifacts that is the information about the system, tiers and servers; and the second step is to define the tasks to be done to configure the server that is the command and action.

Figure 5. Multi-Tier System Deployment

5.1. Run Time Aspect

After getting the SDF, the application starts a workflow to handle some events. The event may be startup, shut-down, scaling-up or scaling-down. When the application encounters the startup event on a server, it invokes the startup sequence and goes through a set of predefined states until the server is started. After starting the server post-startup commands are executed. The post-startup commands could be any arbitrary shell-script to be run on the server just started. User can invoke the custom-event from the UI manually by pressing the events button; if the handler for the event is a scaling activity or a shutting-down activity the corresponding actions will be performed by the application.

5.2. Start-up

When the application encounters the startup event on a server defined in the input Multi-Tier SDF, it invokes the startup sequence as it has been defined and goes through a set of predefined states until the server is started. When it is accessible the system is configured by executing some pre-defined post-startup commands. The generic startup activity sequence has a timeout to prevent the corresponding activities running for too long time; which user can customize. The default timeout is 1800 seconds. If a server takes more time than the timeout to startup then it will be automatically discarded. The startup sequence has provision for defining user-defined actions to be invoked after the generic startup activities are performed. For this user needs to add a post-startup element under the startup definition. One sample Startup Script has been shown in Figure 6.

```
<startup timeout="600"/>
  <post-startup>
    .....
  </post-startup>
</startup>
```

Figure 6. Sample Startup Script

5.3. Post-Start-up

Post-startup command is executed just after the server startup. The post-startup section in deployment definition file contains some script. The post-startup command may be any arbitrary shell-script defined by the user. One sample Post-Startup script has been shown in Figure 7.

```
<post-startup>
  <command type="script" name="installjdk.sh">
    <![CDATA[
      #!/bin/sh
      echo jdk is going to be installed.....
      yum install jdk -y
      echo jdk has been installed.....
    ]]>
  </command>
</post-startup>
```

Figure 7. Sample Post-Startup Script

5.4. Scale-up

The system will be deployed based on user defined number of servers. It is possible to scale-up the system manually or automatically. If the CPU load exceeds certain limit, the applications scale-up the system by adding node or server to the system automatically. Also a user can scale it up manually. The scale-up is guided by the workflow specialized to do the needful. Scale-up is limited by the maximum count of servers. It implies that that a workflow will increase the count of servers by a value defined by user.

5.5. Scale-down

Beside scale-up the system can be scale-down automatically or manually by a user. Scale-down is also guided by a workflow defined to it. Scale-down is limited by the minimum count of servers. It implies that a workflow will decrease the count of servers by a value defined by user.

5.6. Shutdown

All servers have their own shutdown sequence with customizable timeout. On timeout the server will be shutdown and the rest of the system will continue. Basically the shutdown event is invoked by the user. Workflow startup and shutdown time is mentioned as shown in Figure 8 below.

```
<workflow>
  <startup timeout="1500"/>
  <shutdown timeout="900"/>
</workflow>
```

Figure 8. Sample Shutdown Scrip

6. Conclusion

In this paper, a Multi-Tier system deployment technique on Hybrid Cloud Environment has been discussed using Workflow. A web interface has been developed; through which user can deploy a Multi-Tier system just once clicking the Deploy button specifying the requirements. User can access private and public cloud through this common interface.

Here a structure of SDF is given where more than one server is defined within a tier, but all the servers will deploy on the same cloud (either in private or public Cloud). In future work, some research will be done to deploy servers on different Cloud Environment defining in same system definition file.

7. References

[1] G. Kappel, S. Rausch-Schott, W. Retschitzegger; *"A Framework for Workflow Management Systems Based on Objects, Rules and Roles";* ACM Computing Surveys Electronic Symposium on Object-Oriented Application Frameworks, 2000.

[2] F. Leymann and D. Roller; *Production workflow: concepts and techniques;* Prentice-Hall, 2000.

[3] W. Aalst; *"Process-Oriented Architectures for Electronic Commerce and Interorganizational Workflow";* Inf. Systems, Vol. 24/8, 1999.

[4] U. Dayal, M. Hsu, R. Ladin, *Business Process Coordination - State of the Art, Trends, and Open Issues;* Proc. of the 27th VLDB Conference, 2001.

[5] Laine, J.M.; Midorikawa, E.T.; *Efficient Strategies for Workload Distribution in Heterogeneous Computing Systems;* Computational Science and Engineering Workshops, 2008. CSEWORKSHOPS '08. 11th IEEE International Conference.

[6] Michael zur Muehlen, *Workflow-based Process Controlling. Foundation, Design, and Application of Workflow-driven Process Information Systems;* Logos Verlag Berlin, 2004, ISBN 3-8325-0388-9.

[7] Verlag Berlin, 2004, ISBN 3-8325-0388-9.K. Karmakar, P. Roy, *Infrastructure Oriented Hybrid Cloud Architecture;* International Journal of Innovations in Engineering & Technology (IJIET), Vol. 3, Issue 1, October 2013

[8] http://opensymphony.com/osworkflow

Appendix

A. Sample Structure of SDF

```
<system>
<name>System1</name>
<tiers>
<tier>
<name>web-tier</name>
<servers>
<server>
<name>web-server</name>
<noOfServer>1</noOfServer>
<imageIdentifier>emi-D7621489</imageIdentifier>
<keypair>suvo-key</keypair>
<kernelImage>eki-8A131363</kernelImage>
<ramdiskImage>eri-E7CE147B</ramdiskImage>
<instanceType>m1.large</instanceType>
<securityGroup>default</securityGroup>
<availabilityZone>SMCC-JU-1</availabilityZone>
<workflow>
<startup timeout="600">
<post-startup>
</post-startup>
</startup>
<shutdown/>
</workflow>
</server>
</servers>
</tier>
<tier>
<name>app-tier</name>
<servers>
<server>
<name>app-server1</name>
<noOfServer>1</noOfServer>
<imageIdentifier>emi-A10213AB</imageIdentifier>
<keypair>my-key</keypair>
<kernelImage>eki-12731539</kernelImage>
<ramdiskImage>eri-0CC41526</ramdiskImage>
<instanceType>m1.small</instanceType>
<securityGroup>kk-sec</securityGroup>
<availabilityZone>SMCC-JU-1</availabilityZone>
<workflow>
<startup timeout="600">
<post-startup>
</post-startup>
</startup>
<shutdown/>
</workflow>
</server>
<server>
<name>app-server2</name>
<noOfServer>1</noOfServer>
<imageIdentifier>emi-F03A116E</imageIdentifier>
<keypair>suvo-key</keypair>
<kernelImage>eki-EDA914A1</kernelImage>
<ramdiskImage>eri-08F514FE</ramdiskImage>
<instanceType>m1.xlarge</instanceType>
<securityGroup>default</securityGroup>
<availabilityZone>SMCC-JU-2</availabilityZone>
<workflow>
<startup timeout="600">
<post-startup>
<command type="script" name="installjdk.sh">
<![CDATA[
#!/bin/sh
echo jdk is going to be installed.....
yum install jdk -y
echo jdk has been installed.....
]]>
</command>
</post-startup>
</startup>
<shutdown/>
</workflow>
</server>
</servers>
</tier>
<tier>
<name>web-tier</name>
<servers>
<server>
<name>db-server1</name>
<noOfServer>2</noOfServer>
<imageIdentifier>emi-96901396</imageIdentifier>
<keypair>test</keypair>
<kernelImage>eki-0EED1525</kernelImage>
<ramdiskImage>eri-962E134E</ramdiskImage>
<instanceType>m1.large</instanceType>
<securityGroup>default</securityGroup>
<availabilityZone>SMCC-JU-1</availabilityZone>
<workflow>
<startup timeout="600">
<post-startup>
</post-startup>
</startup>
```

<*shutdown/*>
</*workflow*>
</*server*>
</*servers*>
</*tier*>
</*tiers*>
</*system*>