CHAPTER 5

INTEGRATION OF PROPOSED ARCHITECTURE
WITH CARE RESOURCE BROKER

5.1 OUTLINE

This chapter describes the integration of proposed architecture with our CRB. The CRB has been known as Grid Virtualization Framework (GVF) due to the fact that it supports both grid and virtual resources. For that, a brief introduction to CRB is given first, and then the integration and implementation of proposed architecture with CRB is explained.

5.2 INTEGRATION OF PROPOSED ARCHITECTURE WITH CARE RESOURCE BROKER

The CRB has been developed and implemented by the CARE, Anna University, India. It hides the complexity of deploying clusters and additional software from the users, makes life easy for grid users and application developers. The salient features of CRB are the dynamic creation and deployment of virtual clusters, dynamic creation of grid environment by installing globus middleware on the fly in the virtual cluster, and dynamic preparation of software environment in that virtual cluster thereby creating a complete execution environment dynamically without human intervention.

The CRB (Thamarai Selvi Somasundaram et al, 2010) follows the resource oriented Meta-scheduling model aimed at optimizing the utilization of resources in grid environment. The layered architecture of grid virtualization with CRB that defines various components for scheduling of
jobs to physical as well as virtualized grid resources is shown in Figure.5.1. Fabric layer defines the physical resources participating in the grid for collaborative problem solving. These resources spans across organization domains and geographical boundaries. They are autonomous and have their own administrative policies. However, they allow grid customers to deploy Virtual Clusters in the resources along with the required application specific environment. Resource and Connectivity Layer is built with grid middleware that supports secure communication between physical grid resources using X509 digital certificates. The services defined in this layer are responsible for managing the execution of application in physical resources as well as monitoring. Collective Layer is responsible for resource aggregation as well as scheduling of job onto resources. It performs discovery of appropriate resource for virtual cluster deployment, manages and monitors the virtual clusters as well as other physical resources across the grid infrastructure. A new protocol called Virtual Resource Management Protocol (VRMP) has been proposed for deployment of virtual cluster and the preparation of grid enabled execution environment on the fly. It implements appropriate mechanisms and functionalities to aggregate virtual resources information along with physical resources information. It initiates virtual clusters creation in a remote physical resource and manages their lifetime. It implements appropriate driver components to interact with the underlying grid middleware installed in the physical resources and also with the virtual clusters deployed in the physical resources.

Application Layer contains user interface for job submission across virtualized grid resources. It allows the user to interact with the underlying collective layer services. It can also include workflow engine that allows composition of several applications and submit to grid for execution. This layer allows users from diverse communities with their applications demanding wider range of requirements to access grid resources through
virtualization technology. Then the next step is to plan the integration of the proposed architecture with the above mentioned CRB. The SLA enabled CRB is shown in Figure 5.1.
Before entering into the flow of SLA enabled CRB, the modifications done to the existing CRB are explained. The PMS is integrated with the host identifier, which is available in the controller component in such a way that it will use the policy information while selecting the resources. The image repository and the SLA creation modules are interconnected with the scheduler module. This leads to negotiation with resources that are selected by the scheduler and also to select the VM image from the repository. The SLA monitoring module integrated with Executor in a way to monitor the SLOs and also with SLA enforcement module to find the violations and invokes the appropriate penalty action. The Virtual Resource Manager (VRM) module is integrated with the hypervisor adapter in such a way that to manage the created VMs.

The CRB accepts the jobs in standard Job Submission Description Language (JSDL) (Job 2005) format. Once the jobs are submitted to the CRB, the host identifier gets all the available resources’ information from host pool. Then it requests the PMS to find out the resources that are available at the ETR of the jobs. This is a two fold process: finding the appropriate policy from a set of policies defined for that particular resource and generates policy matched resources list based on the ETR alone irrespective of the capability. Every resources participating in grid may have more than one usage policies based on their availability. So the PMS gets the resource list and ETR of the jobs from host identifier, and contacts the policy repository (i.e. DB) to get all the policies for each resource in the resource list. It compares the ETR of each job with the availability tag of usage policy. It is mandatory to ensure that the resources are not reserved by other jobs during that time. If that resource is available in that period, then it will be added to the policy matched resources list. This policy matched resources list is send back to the host identifier to identify each resource in the list fall into which region (i.e. plug-in, exact or subsume) using DRS algorithm.
After this step, every submitted job has a set of matched resources (both policy and capability) that are grouped into three regions. Then the scheduling type of each job is decided based on the resource availability in these regions (i.e. which region and which resource in that region). If a job has resources in all the three regions, then it uses a lollipop based selection to decide the region to be considered. If a job has more than one resource in each region, then the selection of resource from single region is done based on the preferences specified by the user. Then the scheduler orders the resources based on their deviation values. If a job has any resources in the exact or plug-in region, then the job could be executed with the help of single physical resource provider and no need to co-allocate more than one resource provider (physical resources) or to create virtual machines. Suppose a job doesn’t have any resources in the exact and plug-in region but it has the resources only in the subsume region. In that case, the proposed approach carefully explore this region and propose some scheduling use cases in such a way that to use the resources in this region to the fullest possible extend. The proposed scheduling use cases are physical co-allocation, virtual co-allocation and virtual cluster creation. The physical co-allocation takes place, when the resources in the subsume region satisfies the software requirements, QoS requirements and hardware requirements except the number of CPUs needed for job execution. In such a scenario, the meta-scheduler co-allocates two or more physical resources and executes the job in the newly created logical resource. The virtual co-allocation scenario is same as physical co-allocation. The only difference is that instead of co-allocating the physical resources, it explores the possibility of creating the virtual machine in the physical resources itself and thereby growing in the number of resources by co-allocating these newly created virtual machines with physical resources. It is the responsibility of the meta-scheduler to choose either the physical co-allocation or virtual co-allocation by considering some metrics such as deadline, budget, bandwidth and delay. Finally, the virtual cluster creation
initiated only when the resources in the subsume region satisfies the hardware (HR) and QoS requirements (QR) but not the software requirements. The scheduling scenarios (SS) that are derived from deviation based resource scheduling algorithm is shown in the Table 5.1.

**Table 5.1 Scheduling scenarios based on the deviation values from DRS**

<table>
<thead>
<tr>
<th>HR</th>
<th>SR</th>
<th>QR</th>
<th>SS</th>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCP</td>
<td>RAM</td>
<td>SS</td>
<td>Speed</td>
<td>OS</td>
</tr>
<tr>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>✗</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

The resources are ordered based on their preferences (highest bandwidth or low latency) and initiate SLA negotiation. At the end of SLA negotiation, the concrete SLA is created and stored in SLA repository. The negotiated resources are reserved so that they should not be considered for another job scheduling. If the scheduling type of a job is single physical, then the job is submitted to the physical cluster whose name is available within the Hostname tag of SLA schema. Once the job reaches the executing resource, then the SLA job monitoring and SLA monitoring services are invoked to monitor against the commitment. In case of any violation, it will be notified to the SLA enforcement service for further action. If it is a physical co-allocation, then it calls the co-allocation service of globus. In case of virtual co-allocation, the additional virtual machines are created over the selected physical resources specified in the hostname tag of SLA and add it to the
existing cluster by invoking Virtual Machine Service (VMS). The hypervisor installed over the selected resource(s) is obtained from MDS4 (Monitoring and Discovery Service). Based on the available hypervisor, it will query the image repository to get the suitable VM image (.img or .vmdk). The images in the repository may be of bundle based or image based. The image based images are those that they have the operating system, dependent software, libraries etc as a single image. The disadvantages of this type of image based images are that it is tedious to add / delete the software available in that image. But the advantage is that they are ready to use component. The bundling overhead is reduced in image based images. But in the bundle based type, the OS, software are available as separate entity. So it has to bundle all the needed combination to make it as single, ready to use image. The advantage of this type of image is the capability to create any number combination of images of our wish. The disadvantage is additional overhead incurred due to bundling on-demand. Then the selected VM image will be transferred from CRB node to selected host with the help of transporter. After the successful transfer of VM image, the scheduler will call the hypervisor adapter with selected VM image name and the selected hostname. Then the hypervisor adapter will initiate the appropriate adapter (Xen or VMware) with this information. Then the hypervisor adapter will establishes a connection with the selected host and add / activate the context information such as IP address, hostname etc to this transferred image.

In case no suitable image found in the image repository, the scheduler modifies the configuration of existing image to meet the request. Once the VM image gets activation, the hypervisor adapter boots the VM and adds it to the existing cluster. Then the scheduler will submit the job in the new hybrid resource. The virtual cluster scheduling type creates the virtual cluster in the committed resources and executes the job in the newly created virtual cluster with the help of Virtual Cluster Service (VCS) and VMS. The
VCS does both the VM creation and cluster set-up. Once the job gets submitted over the resources, the SLA monitoring module starts its thread and monitors the committed SLOs by measuring the runtime parameters of a running job. In order to retrieve the appropriate SLA, it uses the JobID tag as its key. If there is any violation (either provider side or at user side) during the execution of job, it invokes the corresponding penalty action that is specified in the Penalty tag. After the completion of the job, the SLA monitoring thread deletes the apt SLA from SLA repository.

In brief, the details about the integration of PMS, GSMA and DRS with our CRB are explained in this chapter.