Extending Conventional Grid Scheduler for Leasing Resources


*Centre for Advanced Computing Research and Education, Department of Information Technology, Madras Institute of Technology, Anna University Chennai, Ph: +91-9444030027
www.annauniv.edu/care
stselvi@annauniv.edu,balachandar.ra@gmail.com

Abstract—In this paper, we extend conventional grid scheduler to implement resource leasing strategy for mediating resource leases on behalf of the users. The proposed mechanism allows the resource providers to advertise its willingness to provide the hardware resource as lease and also to express lease usage policies. Further, it is also proposed to extend standard job submission description language to express resource lease request.

I. INTRODUCTION

The emergence in virtualization technologies has lead to the concept of resource leases where potential resources are provided to the user on lease basis after creating required execution environment on the fly [1, 2]. Further, virtual machines are completely isolated and hence any ‘malfunctioning’ of virtual machines does not affect the physical host. Due to this fact, resource providers are very much encouraged to provide their resources as resource leases rather than allowing an unknown application to get executed in the physical host as in grid. Thus, it is very evident that cloud resources can also be a part of grid resources that need to be advertised, and brokered for specific lease request. Hence, we argue that, a typical grid scheduler must be able to negotiate across multiple cloud resources in addition to grid resources on behalf of the users to identify suitable resource for his needs. In this paper, we highlight these issues and try to address them by

- Proposing a resource leasing architecture by extending conventional grid scheduling mechanism, and Extending standard job submission description language to support resource lease request.

II. RESOURCE LEASE BROKERING ARCHITECTURE

The grid scheduling layer shown in Fig. 1 defines various components and services needed for typical grid scheduling as well as resource leasing. The Dispatcher accepts jobs from user described using job description language. It implements a job pool in which all job requests are placed and processed. The matchmaker understands the type of request, that is, job submission or resource lease, from the description. If the request is for job submission, it follows conventional grid scheduling approach for application scheduling.

If the request is for resource lease, it then identifies suitable cloud resources matching hardware requirements and sends a list of matched resources to resource leaser which in turn contacts SLA negotiator establishes an SLA with one cloud resource provider.

The SLA negotiator starts matching / negotiating the QoS parameters such as availability, bandwidth, price, etc. with every potential cloud resource provider and select the best match among them. The SLA management architecture proposed in our earlier work [3] can compare the QoS requirements of the users (A) and that offered by the cloud providers (B) to identify different degree of matches namely Exact, in which A is exactly matched with B, Subsume, in which A is overwhelmed by the cloud provider, and Plug in,
in which B is overwhelmed by A. Thus, at the end of the negotiation, a cloud resource is selected for lease.

The leaser then initiates Virtual Resource Service (VRS) in the remote cluster. It obtains context parameters such as IP address, Software and libraries, hostname etc and based on the requirements, and transfers required Virtual Machine (VM) images from image repository to the selected resource. The VM images are pre-configured to take these context parameters on the fly and booted.

Once the resource is ready, the lease component initiates the exchange of secure keys, that is, public keys of both user and virtual resources, and establishes Secure Shell connection between the user and virtual resource. This completes the ‘resource leasing’ process. The ‘session monitor’ now starts monitoring the activities of virtual machines, the committed SLA parameters, and ‘end time’ of the lease. Once, the lease time is over, the session monitor stops the access and release the virtual resource to physical host. It is then followed by Accounting and Billing for the access.

The information service of grid middleware layer is extended to support expression of willingness to provide the resource as ‘lease’ in XML format. This information has been provided to the grid middleware and aggregated at the metascheduler.

III. RESOURCE LEASING STRATEGY

The resource leasing strategy proposed in this architecture extends conventional grid scheduling mechanism, for example, the one found in Gridway metascheduler [3]. In resource leasing, jobs will not submitted by the scheduler but access to the resource will be provided to the user. This allows him to create platform of his interest and execute application in the leased resource. To submit a lease request, conventional JSDL specification needs to be extended to include required elements to specify the type of request and leases as shown in the fig. 2. The Matchmaking module obtains the JSDL request from dispatcher and parses it. If the request type is ‘Resource Lease’, it then contacts the cloud resource registry to obtain the list of cloud resources registered with it. This list is provided to SLA negotiation component.

```xml
<jSDL:Resources>
  <jSDL:CandidateHost>
    <jSDL:HostName>*.mit.au</jSDL:HostName>
    <jSDL:LeaseType>Immediate</jSDL:LeaseType>
    <jSDL:LeaseType>BestEffort</jSDL:LeaseType>
    <jSDL:LeaseType>AdvanceReservationLease</jSDL:LeaseType>
    <jSDL:startTime/></jSDL:LeaseType>
    <jSDL:endTime/></jSDL:endTime>
  </jSDL:Resources>
</jSDL:Resources>

Fig. 2. An extended JSDL template supports resource lease request
```

IV. FURTHER SCOPE AND CONCLUSION

Conventional grid scheduler cannot schedule application to a resource if it does not contain necessary execution environment. The CARE Resource Broker developed by our lab supports conventional grid scheduling as well as creation of virtual clusters and machines in remote resources [5]. CRB handles this scenario by creating virtual cluster in a potential resource and deploys necessary platform, software and libraries necessary for application execution. The resource leasing strategy proposed in this paper is being implemented in CRB to support resource leases.

In this paper, we highlighted the problem in grid computing paradigm in which cloud resources are also connected to grid. We argue the need of brokering component to negotiate across multiple clouds for resource leases. We understand that it has to be scrutinized further to investigate the overhead involved in creation and management of resource leases which would be our immediate concentration.

ACKNOWLEDGEMENT

The authors sincerely thank the Ministry of Communication and Information Technology, Government of India for financially supporting the Centre for Advanced Computing Research and Education of Anna University, India.

REFERENCES