Dynamic Deployment of Custom Execution Environments in Grids

R.S. Montero, E. Huedo and I.M. Llorente

dsa-research.org

Distributed Systems Architecture Research Group
Universidad Complutense de Madrid
Contents

1. Introduction
2. Straightforward Deployment of VMs
3. Dynamic Provisioning of Computing Elements
4. Related Work
5. Conclusions
1. Introduction

Dynamic Deployment of Custom Execution Environments in Grids

The Problem

• Growing heterogeneity (both hardware & software) of grids highly impacts application porting
  – Increase of cost and length of application porting or development cycle (mainly for testing in the great variety of environments)
  – Limitation of the effective number of resources available to a user/application (only sites bound to a given VO will invest the effort to install, configure and maintain custom software configurations)
  – High operational costs of the infrastructure
1. Introduction

Example: XMM-Newton Science Analysis Software (SAS)

- Analysis of the data provided by XMM-Newton
- Frequently released
- Support for several platforms (OS, hardware)
- Strong software requirements (libraries)
- Must be deployed in all resources

- Impose a significant effort
  - System admin staff
  - Developers
  - Users which may need specific versions

The XMM Newton satellite
1. Introduction

*Dynamic Deployment of Custom Execution Environments in Grids*

### Possible Solutions

- **Software-environment configuration systems**
  - Let users define what applications they want
  - Let administrators make applications available to users
  - Example: SoftEnv
  - They do not completely solve any of the previous problems

- **Deployment of software-environment overlays**
  - Deploy custom software configurations in user-space
  - Example: Condor GlideIn (to deploy Condor pools)
  - Software must be installed in user space
  - Compatibility issues

- **Virtual Machine technologies**
  - Natural way to deal with the heterogeneity of the infrastructure
  - Allow partitioning and isolation of physical resources
  - Execution of legacy applications or scientific codes
  - Examples: In-VIGO, VWS
2. Straightforward Deployment of VMs

Dynamic Deployment of Custom Execution Environments in Grids

Main Idea

• Encapsulate a virtual machine in a grid job.
  – Incorporate the functionality of a general purpose metascheduler
  – Do not need new middleware
  – The underlying LRMS is not aware of the nature of the job
  – Only suitable to medium/coarse grained HTC applications.

• Generalization of previous overlays for Grids:
  – Condor GlideIn
  – GridWay & BOINC
2. Straightforward Deployment of VMs

Dynamic Deployment of Custom Execution Environments in Grids

The GridWay Metascheduler

- Advanced scheduling
- Different application profiles
- Fault detection & recovery
- Job execution management
  - Prolog (stage-in)
  - Wrapper (execution)
  - Epilog (stage-out)
2. Straightforward Deployment of VMs

Dynamic Deployment of Custom Execution Environments in Grids

XMM-Newton Science Archive (XSA) → VM Image Repository

Front-end

GridFTP

SAS

Virtual WN

Wrapper

GridWay

GridFTP

GRAM

LRMS

Client Machine

Virtual WN

Worker Nodes

3. Epilog (Stage-out)

2.1 Stage-in to virtual WN
2.2 Execution in the virtual WN
2.3 Stage-out to cluster FS
2. Straightforward Deployment of VMs

Dynamic Deployment of Custom Execution Environments in Grids

Experiments

- XMM-Newton SAS application
- Overhead analysis

Testbed Characteristics

<table>
<thead>
<tr>
<th>Host</th>
<th>CPU</th>
<th>Memory</th>
<th>OS</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>ursa</td>
<td>P4 3.2GHz</td>
<td>512MB</td>
<td>FC 4</td>
<td>GW</td>
</tr>
<tr>
<td>draco</td>
<td>P4 3.2GHz</td>
<td>512MB</td>
<td>Etch</td>
<td>GT4, PBS</td>
</tr>
<tr>
<td>draco WN</td>
<td>P4 3.2GHz</td>
<td>2GB</td>
<td>Etch</td>
<td>Xen3.0</td>
</tr>
</tbody>
</table>
2. Straightforward Deployment of VMs

Dynamic Deployment of Custom Execution Environments in Grids

Overhead Analysis

![Chart showing overhead analysis of VM deployment.

- Without VM: 150 seconds
- Persistent VMs: 200 seconds
- Save Restore: 250 seconds
- Start Stop: 300 seconds

Legend:
- Yellow: Boot
- Purple: Restore VM state
- Pink: Save VM state
- Brown: Shutdown
- Blue: Execution]
Main Idea

- New infrastructure layer, separating Resource Provisioning from Job Management
- Seamless integration with the existing middleware stacks
- Completely transparent to the computing service and so end users
3. Dynamic Provisioning of Computing Elements

Dynamic Deployment of Custom Execution Environments in Grids

Features

User Requests
- Typical LRMS interface
- Virtualization overhead

Cluster Frontend

Distributed Virtualizer

VMM

Virtualized cluster nodes

Dedicated cluster nodes

Cluster Nodes
3. Dynamic Provisioning of Computing Elements

Dynamic Deployment of Custom Execution Environments in Grids

Features

Cluster Consolidation
- Multiple worker nodes in a single resource
- Dynamic provision rules (infr. adaptation)
- VMM functionality (e.g. live migration)

Cluster Frontend

Virtualized cluster nodes

Distributed Virtualizer

VMM

Dedicated cluster nodes

Cluster Nodes
3. Dynamic Provisioning of Computing Elements

Dynamic Deployment of Custom Execution Environments in Grids

Features

Cluster Partitioning
- Performance partitioning (dedicated workernodes)
- Isolate cluster workload
- Dedicated HA partitions

Cluster Frontend

Virtualized cluster nodes

Distributed Virtualizer

VMM

Dedicated cluster nodes

Cluster Nodes
3. Dynamic Provisioning of Computing Elements

Dynamic Deployment of Custom Execution Environments in Grids

Features

Heterogenous Workloads
- Dynamic provision of cluster configurations
- Example: on-demand VO workernodes in Grids

Cluster Frontend

Virtualized cluster nodes

Distributed Virtualizer

Dedicated SGE nodes

Cluster Nodes
3. Dynamic Provisioning of Computing Elements

Dynamic Deployment of Custom Execution Environments in Grids

Grid Integration

- Unmodified Grid applications
- Grid interfaces preserved (DRMAA...)

Applications

- Virtual resources are exposed by GT
- Dynamic scheduling
- Fault detection & recovery

Grid Middleware Layer

- WN images registers to a different queue

Computing Service Layer

- VO specific appliances for the WNs
- Coexist with other services

Infrastructure Layer

Cluster Frontend (SGE)

OpenNebula

GridWay

MDS  GRAM  GridFTP
3. Dynamic Provisioning of Computing Elements

Dynamic Deployment of Custom Execution Environments in Grids

Grid Integration

- Adapts the Grid infrastructure to its workload
- WN deployment policies (e.g. VO shares)
- Fault detection & recovery

GridWay

Infrastructure Manager

MDS | GRAM | GridFTP

Cluster Frontend (SGE)

OpenNebula

VWS

VO Appliance repository
3. Dynamic Provisioning of Computing Elements

Dynamic Deployment of Custom Execution Environments in Grids

Experiments

- Interaction between each component
- Overhead induced by each component

Testbed Characteristics

<table>
<thead>
<tr>
<th>Host</th>
<th>CPU</th>
<th>Memory</th>
<th>OS</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCM</td>
<td>P4 3.2GHz</td>
<td>1GB</td>
<td>Etch</td>
<td>GT, SGE, VWS</td>
</tr>
<tr>
<td>ESA</td>
<td>Xeon 2.2GHz</td>
<td>2GB</td>
<td>FC6</td>
<td>GT, SGE, VWS</td>
</tr>
<tr>
<td>Manager</td>
<td>P4 3.2GHz</td>
<td>2GB</td>
<td>Etch</td>
<td>GW, GT</td>
</tr>
</tbody>
</table>
3. Dynamic Provisioning of Computing Elements

Dynamic Deployment of Custom Execution Environments in Grids

Overhead Analysis

GridWay

Infrastructure Manager

170 s

90 s

Cluster Frontend (SGE)

OpenNebula

VMM

VO Appliance repository

MDS

GRAM

GridFTP

~10% of overhead for computational tasks

1 2 sec

2 sec

18 s

2 sec

96 s

3

4

5

6
3. Dynamic Provisioning of Computing Elements

Dynamic Deployment of Custom Execution Environments in Grids

Overhead Analysis

ShUTDOWN
- Same steps
- Similar overhead
- Metascheduler must be able to recover from failure

~10% of overhead for computational tasks
4. Related Work

Dynamic Deployment of Custom Execution Environments in Grids

Renewed Interest on Virtualization Technologies

- **COD** (Cluster on Demand) is a cluster management software
- **Edge Services** uses VWS to deploy VO-dedicated servers
- **In-VIGO** uses VMs to deploy different middleware stacks
- **Amazon EC2** (Elastic Computing Cloud) provides a remote VM execution environment through a simple WS interface
- And many more…
5. Conclusions

*Dynamic Deployment of Custom Execution Environments in Grids*

**Grids and Virtual Machines**

- Both alternatives
  - Reduce application porting times (mainly testing time)
  - Increase the effective number of resources available to a user/application
  - Reduce the operational costs of the infrastructure (simple on-demand provision of custom configurations)
- Straightforward deployment of VMs
  - Almost ready to work on existing infrastructures, with limited overhead for some deployments
  - Does not fully exploit virtualization
  - Limited to medium to coarse grained batch applications
- **Dynamic provisioning of computing elements**
  - New infrastructure layer, separating Resource Provisioning from Job Management
  - Dynamically adapt the infrastructure to support different VOs
  - Seamless integration of remote providers (Amazon EC2, VWS…)
  - Implement different VO policies to adapt the infrastructure (future)
THANK YOU FOR YOUR ATTENTION!!!
More info, downloads, mailing lists at
www.opennebula.org

This work is partially funded by the “RESERVOIR– Resources and Services
Virtualization without Barriers” project
EU grant agreement 215605

www.reservoir-fp7.eu/