Enforcing SLAs in Scientific Clouds

Oliver Niehörster, André Brinkmann, Gregor Fels, Jens Krüger, and Jens Simon
Software as a Service SaaS

- On demand, pay-as-you-go
- Consume software like receiving energy
- Service Level Agreement SLAs define a contract between provider and consumer
- Service Level Objectives SLOs define QoS requirements inside SLA
- Business examples:
  - Word Processor, CRM, CAD, Spreadsheet, ...
### Challenges

<table>
<thead>
<tr>
<th>scientific SW vs. business SW</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>One time jobs</strong></td>
</tr>
<tr>
<td><strong>SLO</strong>: performance or deadline</td>
</tr>
<tr>
<td>Performance depends on job inputs</td>
</tr>
<tr>
<td>measuring the QoS is difficult</td>
</tr>
</tbody>
</table>

A cloud is ...
- private or public,
- a blackbox,
- used by many users.

Enforce SLAs in a stochastic environment, where VMs ...
- influence each other,
- compete against resources.
**Sensors and Actuators**

- **Sensors:**
  - read environmental information
  - monitoring data
  - read log files
  - run benchmarks
  - application specific

- **Actuators:**
  - influence the environment
  - create, configure, setup, change, terminate instances
  - are application specific (especially setup and configuration)
Cost Estimation (CE)

- **Inputs:** job data, SLO
- **Outputs:**
  - cost estimate
  - minimal resources (if SLO is feasible)
- **Possible approaches:**
  - heuristic based on previous job runs
    \[ h(\text{job inputs}) = \text{needed resources} \]
  - pilot runs: do a search with short representative jobs
• One agent per job that has to enforce SLOs

• Skills:
  – Determine the “optimal” provisioning that fulfills the SLO
  – Can change node type
  – Can change number of nodes

• Behavior:
  – Read sensor data
  – Assess QoS
  – Correct provisioning with the actuators
• Reflex agent
• Fuzzy control system
• “soft” SLO as performance contract

IF under-provisioned THEN faster
IF proper-provisioned THEN no change
IF over-provisioned THEN slower
Gromacs as a Service (GraaS)

- Molecular dynamics package designed for biomolecular systems
- SLOs: deadlines
- Sensor:
  - Reads Gromac's time estimates from stdout
- Actuators:
  - Adding and removing of nodes
  - Reconfiguration of the applications (assure that #MPI processes = #vCores)
  - Configure and restart via checkpoints
  - Change type first, then change #VMs
- Cost Estimation:
  - Binary search
  - Pilot runs with the original job data but with fewer iterations
Evaluation

- Eucalyptus
- EC2 interface
- XEN nodes
- \#vCPUs ≤ \#pCores
- Change instance type
  - Ballooning
  - CPU hotplug

- Like the private cloud
- ESX nodes with Infiniband HCAs

- Public cloud
- m1.large (4 EC2 CUs) in us-east-1a
- Ubuntu instances
- Just horizontal scaling
HPC Cloud

- Performance evaluation with Gromacs (DPPC in water)

![Graph showing performance evaluation with Gromacs (DPPC in water). The graph plots the number of MPI processes against the time per core, with different lines representing different configurations: 4C-VM, 2C-VM, 8C-IB, 4C-GbE. The x-axis represents the number of MPI processes ranging from 1 to 16, and the y-axis represents the time per core in nanoseconds.]
Evaluation

- Private Cloud
- Empty: no noise
- Job: *DPPC in water, 12.500 iterations*

<table>
<thead>
<tr>
<th>Deadline [sec]</th>
<th>CE time [sec]</th>
<th>CE #nodes</th>
<th>real end [sec]</th>
<th>real costs [credits]</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>1508</td>
<td>1</td>
<td>1859</td>
<td>84</td>
</tr>
<tr>
<td>1800</td>
<td>1508</td>
<td>1</td>
<td>1614</td>
<td>77</td>
</tr>
<tr>
<td>1600</td>
<td>1500</td>
<td>1</td>
<td>1463</td>
<td>73</td>
</tr>
<tr>
<td>1400</td>
<td>1240</td>
<td>3</td>
<td>1275</td>
<td>130</td>
</tr>
<tr>
<td>1200</td>
<td>1210</td>
<td>3</td>
<td>1142</td>
<td>171</td>
</tr>
<tr>
<td>1000</td>
<td>1044</td>
<td>4</td>
<td>1001</td>
<td>200</td>
</tr>
<tr>
<td>800</td>
<td>812</td>
<td>6</td>
<td>802</td>
<td>240</td>
</tr>
</tbody>
</table>
Evaluation

- Two jobs in a noisy cloud
- Synthetic disk, I/O, and memory load on the hypervisors
Evaluation

- Two jobs in a noisy cloud
- Synthetic disk, I/O, and memory load on the hypervisors
Evaluation

• EC2 observations:
  – Performance of one instance relative stable
  – But differences between identical instances

• Starting with a faulty CE
• **EC2 observations:**
  – Performance of one instance relatively stable
  – But differences between identical instances

• **Starting with a faulty CE**
Summary and Future Work

• **Scientific SaaS with an easy-to-use website**
  – Works with applications that are malleable
  – Provides an interface for QoS measuring
  – Allows a cost estimation

• **Agents:**
  – Application specific
  – Control the jobs
  – Enforce SLA
  – Cost Estimation

• **Future Work:**
  – More applications
  – Smarter agents, who cooperate instead of compete
  – Handle “finite” clouds

• **Evaluation:**
  – GraaS on three different clouds: private, private HPC, EC2
  – Benchmarks of the HPC cloud
Thank you!