Energy-Aware Scheduling in Virtualized Datacenters

Íñigo Goiri, Ferran Julià, Ramón Nou, Josep Ll. Berral, Jordi Guitart and Jordi Torres

Universitat Politècnica de Catalunya and Barcelona Supercomputing Center (BSC)
Energy consumption is a large cost in datacenters
- Server energy consumption: 11%
- PUE overhead: 11%

Virtualization is used to
- Consolidate tasks in the same server
- Save energy and reduce management complexity

Execute HPC tasks on top
- SLAs based on deadlines
Virtualization adds overheads
  - Creation time
  - Migration
  - Disk management

Aggressive consolidation for saving energy
  - May incur in performance loss

Delay finish time of HPC tasks
  - SLA violation → pay penalty
New scheduling policy: score-based
- Focused on running HPC jobs
- Reduce energy consumption
- Manage virtualization overheads
- Reduce management complexity
- Reduce SLA violations
Score-based scheduling

Scheduling algorithm

- Decide where to run a VM dynamically
  - Evaluate every VM allocation in every server
- Give a score to each allocation
  - Aggregation of parameters
  - Lower score is better
- Model datacenter scheduling as a matrix: \( vm \times host \)
  - Each cell represents the cost of allocating a \( vm \) in a \( host \)
  - \( N \): number of virtual machines
  - \( M \): number of active hosts
  - 1 special host: virtual queue
  - Matrix size: \( N \times (M + 1) \)
- Find scheduling with global lower score
M servers

*VM*$_1$ submitted to the datacenter

- Scheduled to *$H_1$*

<table>
<thead>
<tr>
<th></th>
<th>$VM_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_1$</td>
<td>S(1,1)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$H_M$</td>
<td>S(M,1)</td>
</tr>
<tr>
<td>$H_V$</td>
<td>-</td>
</tr>
</tbody>
</table>
Score-based scheduling

Scheduling algorithm

- $VM_1$ running on $H_1$
- $VM_2$ submitted to the datacenter
  - Scheduled to $H_M$

<table>
<thead>
<tr>
<th></th>
<th>$VM_1$</th>
<th>$VM_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_1$</td>
<td>S(1,1)</td>
<td>S(1,2)</td>
</tr>
<tr>
<td>$H_M$</td>
<td>S(M,1)</td>
<td>S(M,2)</td>
</tr>
<tr>
<td>$H_V$</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
After $N$ VMs submitted

<table>
<thead>
<tr>
<th>$H_1$</th>
<th>$S(1,1)$</th>
<th>$S(1,2)$</th>
<th>$S(1,3)$</th>
<th>$S(1,4)$</th>
<th>...</th>
<th>$S(1,N)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$H_M$</td>
<td>$S(M,1)$</td>
<td>$S(M,2)$</td>
<td>$S(M,3)$</td>
<td>$S(M,4)$</td>
<td>...</td>
<td>$S(M,N)$</td>
</tr>
<tr>
<td>$H_V$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Score-based scheduling

Scheduling algorithm
Score-based scheduling

Calculate score

- Score of a tentative allocation of virtual machine VM in server H
- Aggregation of parameters
  - Requirements (booleans)
  - Resources (booleans)
  - Virtualization overhead (time)
  - Power (watts)
  - ...

\[ \text{Score} = \text{weight}_1 \cdot \text{requirements} + \text{weight}_2 \cdot \text{resources} + \ldots \]

- Lower score, better allocation
  - Impossible allocations, \( \infty \) score
Score-based scheduling
Score calculation: Hardware, software, and resource requirements

- If the host cannot fulfill the VM requirements:
  - Lacks required hardware: number of CPUs, disk...
  - Lacks required software
  - Lacks required hypervisor
  - $\infty$ score

- If the host does not have enough free resources:
  - Not enough CPU, memory...
  - $\infty$ score
Score-based scheduling
Score calculation: Virtualization overhead

- Overhead introduced by virtualization management
  - Time to create the VM
  - Time to migrate the VM
- Avoid operating on VMs undergoing migrations
- Minimize concurrent operations
  - Interfere with other actions and make actions take longer
  - Creating two VMs at the same time is slower
Score-based scheduling

Score calculation: Energy efficiency and others

- Estimate energy usage
  - Reward almost full servers: low score
  - Penalize empty servers: high score
- Other parameters which can be added:
  - Fault tolerance
  - SLA enforcement
Matrix solving
System modeled as a matrix

- Put all the scores in the matrix:
  - Bold indicates current allocation
  - VM₄ cannot be executed and it is in the queue

<table>
<thead>
<tr>
<th></th>
<th>VM₁</th>
<th>VM₂</th>
<th>VM₃</th>
<th>VM₄</th>
<th>...</th>
<th>VMₙ</th>
</tr>
</thead>
<tbody>
<tr>
<td>H₁</td>
<td>15.2</td>
<td>15.2</td>
<td>∞</td>
<td>15.2</td>
<td></td>
<td>10.0</td>
</tr>
<tr>
<td>H₂</td>
<td>∞</td>
<td>7.8</td>
<td>7.8</td>
<td>7.8</td>
<td></td>
<td>∞</td>
</tr>
<tr>
<td>H₃</td>
<td>10.3</td>
<td>10.3</td>
<td>∞</td>
<td>10.3</td>
<td></td>
<td>10.5</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hₘ</td>
<td>11.0</td>
<td>∞</td>
<td>11.0</td>
<td>11.0</td>
<td></td>
<td>∞</td>
</tr>
<tr>
<td>Hᵥ</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td></td>
<td>∞</td>
</tr>
</tbody>
</table>
Matrix solving
Solving scheduling

- Calculate scores of the current allocation
- Optimize matrix to get lower scores for VM allocations
  - Hill climbing
- Apply changes to the system
  - Create VM
  - Migrate VM between nodes
  - Keep VMs that cannot be executed in the queue
  - Apply turn on/off policy
Energy savings

Turn on/off policy

• Turn on/off approach: save energy
  • Use two thresholds
  • Turn off idle servers
    • Turn off servers as soon as they are not used
  • Turn on new machines if they are required
    • Wait until machines are required

• More consolidation
  • More energy savings
  • Lower performance
Simulated environment

**Workload**
- One week of Grid 5000 workload
- \( \sim 2000 \) tasks with an average of \( \sim 5000 \) seconds per task

100 virtualized hosts

**Metrics:** energy consumption, client satisfaction
Evaluation

Power simulator

- Simulate nodes with different features
  - Fast and reproducible results

Schedulers
- Random
- Round robin
- Backfilling
- Dynamic backfilling
- Score-based

Workload | Simulator | Output
---|---|---

Íñigo Goiri (igoiri@ac.upc.edu)  Cluster 2010, September 21st 2010
Evaluation

Energy consumption vs. SLA fulfillment trade-off

- More aggressive consolidation (left part)
  - More energy savings
  - Fulfill fewer SLAs
- Less aggressive consolidation (right part)
  - Less energy savings
  - Fulfill more SLAs

(a) Client satisfaction

(b) Power consumption

Íñigo Goiri (igoiri@ac.upc.edu)
Evaluation

Static allocation

- **Metrics**
  - Normalized average power to the best policy
  - Client satisfaction

- **Static scheduling policies**
  - Do not reschedule: no migration

---

**Diagram:**

- **Random**
- **Round Robin**
- **Backfilling**
- **Score simple**
- **Score**

- **Y-axis:** %
- **X-axis:** Policies

- **Legend:**
  - Red: Power (kW)
  - Blue: SLA (%)
Add migration capability to scheduling
- Consolidation is increased
- Score-based gets better results
Comparing energy usage for different scheduling policies
Conclusions and future work

- Improve energy efficiency
- Deal with virtualization overheads
  - More SLAs are fulfilled
- Intuitive formulation
- Easy to extend

Future work
- Add economic model
  - Move from score to economic units
- Evaluate reliability
- Extend to other applications with different SLAs: services, transactional
Energy-Aware Scheduling in Virtualized Datacenters

Íñigo Goiri, Ferran Julià, Ramón Nou, Josep Ll. Berral, Jordi Guitart and Jordi Torres

Universitat Politècnica de Catalunya and Barcelona Supercomputing Center (BSC)