List scheduling heuristics for virtual machine mapping in cloud systems

S. Nesmachnow
S. Iturriaga
Universidad de la República
Uruguay

B. Dorronsoro
E.-G. Talbi
INRIA/University of Lille
France

P. Bouvry
University of Luxembourg
Luxembourg
Outline

1. Motivations and Contributions
2. The Virtual Machine Mapping Problem
3. List Scheduling Algorithms
4. Experiments
5. Conclusions and Future Work
1. Motivations and Contributions
2. The Virtual Machine Mapping Problem
3. List Scheduling Algorithms
4. Experiments
5. Conclusions and Future Work
Motivations

- **Cloud computing**
  - Novel computing paradigm
  - Cloud service provisioning models: IaaS, PaaS, SaaS
  - VMs that can be easily allocated and deallocated
  - Elasticity, flexibility, seemingly infinity of resources, etc.

- **Broker:** intermediary entity between cloud providers and users
  - Finding the best deal
  - ‘Cloudifying’ applications

- **New business model for cloud**
  - Book reserved instances (RI) on a number of cloud providers
    - Low investment
  - Sublet them to its customer as on-demand resources
    - 20% cheaper than the price cloud providers offer
  - Overloaded situations: cloud bursting
Contributions

- Definition of the novel business model
- Formulation of the optimization problem that arises
- Resolution of the problem with 8 heuristics
- Generation of a novel benchmark
  - 400 instances
  - Diverse workloads and scenarios
  - Real pricing data (AWS and Azure)
Outline

1. Motivations and Contributions
2. The Virtual Machine Mapping Problem
3. List Scheduling Algorithms
4. Experiments
5. Conclusions and Future Work
The VMMP Problem

- Allocate all customers VMs requests into the available RIs
The VMMP Problem

- Allocate all customers VMs requests into the available RIs
The VMMP Problem

\[
\begin{align*}
\text{max} & \quad \sum_{j=1}^{j=m} \left( \sum_{i: f(v_i) = b_j} (p(BF(v_i)) - C(b_j)) \times T(v_i) \right) \\
\text{subject to} & \quad M(v_i) \leq M(b_j), \quad P(v_i) \leq P(b_j) \\
& \quad S(v_i) \leq S(b_j), \quad nc(v_i) \leq nc(b_j) \\
& \quad \sum_{h: ST(v_h) > D(v_h)} \left( p(BF(v_h)) - COD(BF(v_h)) \right) \times T(v_h) \quad \text{Cost of deadline violations handling}
\end{align*}
\]
Outline

1. Motivations and Contributions
2. The Virtual Machine Mapping Problem
3. List Scheduling Algorithms
4. Experiments
5. Conclusions and Future Work
List Scheduling Algorithms

- Best fit resource (BFR): assigns every VM to its most suitable RI
- Earliest finish time (EFT): VMs that finish earlier first
- Lower gap first (LGF): VMs with tightest deadlines first
- Shortest task first (STF): VMs with shortest execution time first
- Earliest deadline first (EDF): VMs with earliest deadlines first (arrival time is not taken into account)
- Cheapest instance (CI): VMs are assigned to the cheapest RI that can execute it, in a FIFO
- Max profit (MaxP): VMs that provide higher profit first
- Shortest request to cheapest instance (SRCI): Shortest VMs are first assigned to the cheapest instance that can execute them
Outline

1. Motivations and Contributions
2. The Virtual Machine Mapping Problem
3. List Scheduling Algorithms
4. Experiments
5. Conclusions and Future Work
Experiments

- 20 workload files (information on VMs request)
  - Batches of 50, 100, 200, and 400 VMs
- 20 scenario files (information on available RIs)
  - 10, 20, 30, and 50 RIs (AWS and Azure data)
- 8 different kinds of VMs
- Available online
- Pricing: 20% cheaper than the cloud provider price

<table>
<thead>
<tr>
<th>#</th>
<th>VM id</th>
<th>provider</th>
<th>memory</th>
<th>storage</th>
<th>proc.</th>
<th>nc</th>
<th>price</th>
<th>C</th>
<th>COD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>m1.small</td>
<td>Amazon</td>
<td>1.7 GB</td>
<td>160 GB</td>
<td>1.0 GHz</td>
<td>1</td>
<td>0.048</td>
<td>0.027</td>
<td>0.06</td>
</tr>
<tr>
<td>2</td>
<td>m1.medium</td>
<td>Amazon</td>
<td>3.75 GB</td>
<td>410 GB</td>
<td>2.0 GHz</td>
<td>2</td>
<td>0.096</td>
<td>0.054</td>
<td>0.12</td>
</tr>
<tr>
<td>3</td>
<td>A2.medium</td>
<td>Azure</td>
<td>3.5 GB</td>
<td>489 GB</td>
<td>1.6 GHz</td>
<td>2</td>
<td>0.096</td>
<td>0.09</td>
<td>0.12</td>
</tr>
<tr>
<td>4</td>
<td>m1.large</td>
<td>Amazon</td>
<td>7.5 GB</td>
<td>850 GB</td>
<td>2.0 GHz</td>
<td>4</td>
<td>0.192</td>
<td>0.108</td>
<td>0.24</td>
</tr>
<tr>
<td>5</td>
<td>m2.xlarge</td>
<td>Amazon</td>
<td>17.1 GB</td>
<td>420 GB</td>
<td>3.25 GHz</td>
<td>2</td>
<td>0.192</td>
<td>0.136</td>
<td>0.24</td>
</tr>
<tr>
<td>6</td>
<td>A3.large</td>
<td>Azure</td>
<td>7.0 GB</td>
<td>999 GB</td>
<td>1.6 GHz</td>
<td>4</td>
<td>0.328</td>
<td>0.18</td>
<td>0.41</td>
</tr>
<tr>
<td>7</td>
<td>c1.xlarge</td>
<td>Amazon</td>
<td>7.0 GB</td>
<td>1690 GB</td>
<td>2.5 GHz</td>
<td>8</td>
<td>0.384</td>
<td>0.316</td>
<td>0.48</td>
</tr>
<tr>
<td>8</td>
<td>A4.xlarge</td>
<td>Azure</td>
<td>14.0 GB</td>
<td>2039 GB</td>
<td>1.6 GHz</td>
<td>8</td>
<td>0.464</td>
<td>0.36</td>
<td>0.58</td>
</tr>
</tbody>
</table>

5.2 Development and execution platform
The proposed heuristics were implemented in C, using stdlib and GNU gcc. The experiments were performed on a Xeon E5430, 2.66 GHz, 8GB RAM, and CentOS Linux 5.2 from Cluster FING (http://www.fing.edu.uy/cluster).
Results: Broker Profit
Results: Broker Profit

- Friedman test

<table>
<thead>
<tr>
<th>heuristic</th>
<th>batch dimension (n)</th>
<th>overall</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>BFR</td>
<td>5.35</td>
<td>6.34</td>
</tr>
<tr>
<td>EFT</td>
<td>6.61</td>
<td>5.81</td>
</tr>
<tr>
<td>LGF</td>
<td>6.76</td>
<td>6.44</td>
</tr>
<tr>
<td>STF</td>
<td>2.99</td>
<td>3.36</td>
</tr>
<tr>
<td>STF</td>
<td>2.99</td>
<td>3.36</td>
</tr>
<tr>
<td>STF</td>
<td>2.99</td>
<td>3.36</td>
</tr>
<tr>
<td>STF</td>
<td>2.99</td>
<td>3.36</td>
</tr>
<tr>
<td>STF</td>
<td>2.99</td>
<td>3.36</td>
</tr>
<tr>
<td>STF</td>
<td>2.99</td>
<td>3.36</td>
</tr>
<tr>
<td>SRCI</td>
<td>1.71</td>
<td>1.16</td>
</tr>
<tr>
<td>MaxP</td>
<td>3.02</td>
<td>3.14</td>
</tr>
</tbody>
</table>

Table 2 shows that SRCI is consistently the heuristic providing the highest benefit to the broker: it got the lowest average rank for every batch dimension. We can see that, aside SRCI, the heuristics that target the profit optimization (namely, CI and MaxP) are generally outperformed by STF in all problem sizes. Both SRCI and STF assign first the shortest tasks, which arises a proper policy. Finally, EDF, EFT, LGF, and BFR performed the worst in all cases.

The profit results are reported in Table 3. The GAP metric is the relative difference between the profit computed using each heuristic and the best result for each problem instance. Row #1 indicates how many times the corresponding heuristic performed the best (i.e., the number one) regarding the profit value. Some heuristics report negative profit values for several of the tested instances. Particularly, this happens for those instances with a high ratio between the VM requests and the available RIs, for which the available RIs are not enough.
Results: Broker Profit

- **batch dimension = 50**
- **batch dimension = 100**
- **batch dimension = 200**
- **batch dimension = 400**

Heuristic performance comparison:
- **BFR**
- **EFT**
- **LGF**
- **STF**
- **EDF**
- **CI**
- **SRCI**
- **MaxP**
Results: Violated Requests

- batch dimension = 50
- batch dimension = 100
- batch dimension = 200
- batch dimension = 400

The chart shows the percentage of violated requests for different heuristics and batch dimensions. Each bar represents a heuristic, and the color indicates the batch dimension.
Outline

1. Motivations and Contributions
2. The Virtual Machine Mapping Problem
3. List Scheduling Algorithms
4. Experiments
5. Conclusions and Future Work
Conclusions & Future Work

• Novel cloud brokering model
  - Reserved nodes are sublet in an on-demand basis
  - Profitable: large difference between on-demand and reserved VMs cost

• VMMP: novel problem to plan the resources utilization
  - VMs requests must be mapped into RIs, maximizing profit
  - Constraint violations imply profit reduction

• Eight heuristics to solve the problem
  - SRCI outperformed the others

• Future work
  - Use of metaheuristics
  - Consider nesting technology
  - Development techniques for accurately managing the number and kind of RIs
Thank you.