A dynamic optimization model for power and performance management of virtualized clusters

Vinicius Petrucci, Orlando Loques
Univ. Federal Fluminense
Niteroi, Rio de Janeiro, Brasil

Daniel Mossé
Univ. of Pittsburgh, USA
Impact of energy consumption

• Energy crisis
  – High energy costs and demand growing

• Environmental impacts
  – Carbon dioxide emission
  – Global warming

• Economic impacts
  – Resource scarcity leads to higher market prices
  – Clean and renewable sources may have high costs

• **Power-efficiency** is a fundamental concern in today's large server clusters / data-centers
Our solution

• Integrate power and performance management in heterogeneous server clusters
• Virtualized platform targeted at hosting multiple independent web applications
• **Optimization** approach to
  1. dynamically manage the cluster power consumption
  2. meet the application's performance demands
Optimization approach

• The optimization problem:
  – Determine the most power-efficient cluster configuration that can handle a given workload
  – Variant of the bin packing problem

• A cluster configuration is given by
  1. which servers must be active and their respective CPU frequencies
  2. a corresponding mapping of the apps (running on top of VMs) to physical servers
Optimization approach

- Optimization is given by a MIP model solved periodically in a control loop fashion
  1. Solve an optimization problem
  2. Use the solution to configure the cluster

![Diagram showing the optimization process](image)
## Power and performance model

<table>
<thead>
<tr>
<th>Server 1: ampere</th>
<th>Server 4: joule</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU: AMD Athlon(tm) 64 X2 3800+</td>
<td>CPU: AMD Athlon(tm) 64 3500+</td>
</tr>
<tr>
<td>Freq. (MHz)</td>
<td>( P_{\text{busy}} ) (W)</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------</td>
</tr>
<tr>
<td>1000</td>
<td>81.5</td>
</tr>
<tr>
<td>1800</td>
<td>101.8</td>
</tr>
<tr>
<td>2000</td>
<td>109.8</td>
</tr>
<tr>
<td>2200</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Server 2: coulomb</th>
<th>Server 5: ohm</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU: AMD Athlon(tm) 64 3800+</td>
<td>CPU: AMD Athlon(tm) 64 X2 5000+</td>
</tr>
<tr>
<td>Freq. (MHz)</td>
<td>( P_{\text{busy}} ) (W)</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------</td>
</tr>
<tr>
<td>1000</td>
<td>75.2</td>
</tr>
<tr>
<td>1800</td>
<td>89.0</td>
</tr>
<tr>
<td>2000</td>
<td>94.5</td>
</tr>
<tr>
<td>2200</td>
<td>100.9</td>
</tr>
<tr>
<td>2400</td>
<td>107.7</td>
</tr>
<tr>
<td>2600</td>
<td>140.1</td>
</tr>
</tbody>
</table>

Different **power** and **performance levels in heterogeneous server clusters**
# Dynamic Voltage/Frequency Scaling

Power and performance model

<table>
<thead>
<tr>
<th>Server 1: ampere</th>
<th>CPU: AMD Athlon(tm) 64 X2 3800+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freq. (MHz)</td>
<td>( P_{\text{busy}} ) (W)</td>
</tr>
<tr>
<td>1000</td>
<td>81.5</td>
</tr>
<tr>
<td>1800</td>
<td>101.8</td>
</tr>
<tr>
<td>2000</td>
<td>109.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Server 2: coulomb</th>
<th>CPU: AMD Athlon(tm) 64 3800+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freq. (MHz)</td>
<td>( P_{\text{busy}} ) (W)</td>
</tr>
<tr>
<td>1000</td>
<td>75.2</td>
</tr>
<tr>
<td>1800</td>
<td>89.0</td>
</tr>
<tr>
<td>2000</td>
<td>94.5</td>
</tr>
<tr>
<td>2200</td>
<td>100.9</td>
</tr>
<tr>
<td>2400</td>
<td>107.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Server 3: hertz</th>
<th>CPU: AMD Athlon(tm) 64 3800+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freq. (MHz)</td>
<td>( P_{\text{busy}} ) (W)</td>
</tr>
<tr>
<td>1000</td>
<td>71.6</td>
</tr>
<tr>
<td>1800</td>
<td>85.5</td>
</tr>
<tr>
<td>2000</td>
<td>90.7</td>
</tr>
<tr>
<td>2200</td>
<td>96.5</td>
</tr>
<tr>
<td>2400</td>
<td>103.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Server 4: joule</th>
<th>CPU: AMD Athlon(tm) 64 3500+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freq. (MHz)</td>
<td>( P_{\text{busy}} ) (W)</td>
</tr>
<tr>
<td>1000</td>
<td>74.7</td>
</tr>
<tr>
<td>1800</td>
<td>95.7</td>
</tr>
<tr>
<td>2000</td>
<td>103.1</td>
</tr>
<tr>
<td>2200</td>
<td>110.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Server 5: ohm</th>
<th>CPU: AMD Athlon(tm) 64 X2 5000+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freq. (MHz)</td>
<td>( P_{\text{busy}} ) (W)</td>
</tr>
<tr>
<td>1000</td>
<td>82.5</td>
</tr>
<tr>
<td>1800</td>
<td>99.2</td>
</tr>
<tr>
<td>2000</td>
<td>107.3</td>
</tr>
<tr>
<td>2200</td>
<td>116.6</td>
</tr>
<tr>
<td>2400</td>
<td>127.2</td>
</tr>
<tr>
<td>2600</td>
<td>140.1</td>
</tr>
</tbody>
</table>

Dynamic Voltage/Frequency Scaling
### Power and performance model

<table>
<thead>
<tr>
<th>Server 1: ampere</th>
<th>CPU: AMD Athlon(tm) 64 X2 3800+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freq. (MHz)</td>
<td>$P_{busy}$ (W)</td>
</tr>
<tr>
<td>1000</td>
<td>81.5</td>
</tr>
<tr>
<td>1800</td>
<td>101.8</td>
</tr>
<tr>
<td>2000</td>
<td>109.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Server 2: coulomb</th>
<th>CPU: AMD Athlon(tm) 64 3800+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freq. (MHz)</td>
<td>$P_{busy}$ (W)</td>
</tr>
<tr>
<td>1000</td>
<td>75.2</td>
</tr>
<tr>
<td>1800</td>
<td>89.0</td>
</tr>
<tr>
<td>2000</td>
<td>94.5</td>
</tr>
<tr>
<td>2200</td>
<td>100.9</td>
</tr>
<tr>
<td>2400</td>
<td>107.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Server 3: hertz</th>
<th>CPU: AMD Athlon(tm) 64 3800+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freq. (MHz)</td>
<td>$P_{busy}$ (W)</td>
</tr>
<tr>
<td>1000</td>
<td>71.6</td>
</tr>
<tr>
<td>1800</td>
<td>85.5</td>
</tr>
<tr>
<td>2000</td>
<td>90.7</td>
</tr>
<tr>
<td>2200</td>
<td>96.5</td>
</tr>
<tr>
<td>2400</td>
<td>103.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Server 4: joule</th>
<th>CPU: AMD Athlon(tm) 64 3500+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freq. (MHz)</td>
<td>$P_{busy}$ (W)</td>
</tr>
<tr>
<td>1000</td>
<td>74.7</td>
</tr>
<tr>
<td>1800</td>
<td>95.7</td>
</tr>
<tr>
<td>2000</td>
<td>103.1</td>
</tr>
<tr>
<td>2200</td>
<td>110.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Server 5: ohm</th>
<th>CPU: AMD Athlon(tm) 64 X2 5000+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freq. (MHz)</td>
<td>$P_{busy}$ (W)</td>
</tr>
<tr>
<td>1000</td>
<td>82.3</td>
</tr>
<tr>
<td>1800</td>
<td>99.2</td>
</tr>
<tr>
<td>2000</td>
<td>107.3</td>
</tr>
<tr>
<td>2200</td>
<td>116.6</td>
</tr>
<tr>
<td>2400</td>
<td>127.2</td>
</tr>
<tr>
<td>2600</td>
<td>140.1</td>
</tr>
</tbody>
</table>

Server on/off mechanisms (e.g., standby + Wake-on-LAN)
Optimization model

Two ways of mapping app workloads to VMs

(a) every app runs in only one VM instance on a given server,
(b) one given app may run in more than one VM instance, whereas these VMs are balanced among multiple servers
Optimization model

Input variables:

- \( N \) = set of servers in the cluster
- \( F_i \) = set of frequencies for each server \( i \) in \( N \)
- \( M \) = set of apps/services in the cluster
- \( \text{cap}_{ij} \) = capacity (e.g., req/s) of server \( i \) at frequency \( j \)
- \( \text{pb}_{ij}, \text{pi}_{ij} \) = power costs (idle and busy) of server \( i \) at freq. \( j \)
- \( d_k \) = demand (workload) of app \( k \)

Decision variables:

- \( y_{ij} \) = 1 if server \( i \) runs at frequency \( j \), 0 otherwise
- \( x_{ijk} \) = 1 if server \( i \) uses frequency \( j \) to run app \( k \), 0 otherwise
- \( \alpha_{ijk} \) = utilization variable \([0,1]\) of server \( i \) at frequency \( j \)
Optimization model

Problem formulation

Minimize \[ \sum_{i \in N} \sum_{j \in F_i} \alpha_{ij} \cdot pb_{ij} + (y_{ij} - \alpha_{ij}) \cdot pi_{ij} \] (1)

Subject to \[ \sum_{k \in M} d_k \cdot x_{ijk} \leq cap_{ij} \cdot y_{ij} \quad \forall i \in N, \quad \forall j \in F_i \] (2)

\[ \sum_{i \in N} \sum_{j \in F_i} x_{ijk} = 1 \quad \forall k \in M \] (3)

\[ \sum_{j \in F_i} y_{ij} \leq 1 \quad \forall i \in N \] (4)

\[ \alpha_{ij} \leq y_{ij} \quad \forall i \in N, \quad \forall j \in F_i \] (5)

\[ x_{ijk} \in \{0, 1\}, \quad y_{ij} \in \{0, 1\}, \quad \alpha_{ij} \in [0, 1] \]
Optimization model

Minimization of the overall power consumption

Minimize

\[
\sum_{i \in N} \sum_{j \in F_i} \alpha_{ij} \cdot p_{bij} + (y_{ij} - \alpha_{ij}) \cdot p_{ij}
\]  

(1)

Subject to

\[
\sum_{k \in M} d_k \cdot x_{ijk} \leq cap_{ij} \cdot y_{ij} \quad \forall i \in N, \forall j \in F_i
\]  

(2)

\[
\sum_{i \in N} \sum_{j \in F_i} x_{ijk} = 1 \quad \forall k \in M
\]  

(3)

\[
\sum_{j \in F_i} y_{ij} \leq 1 \quad \forall i \in N
\]  

(4)

\[
\alpha_{ij} \leq y_{ij} \quad \forall i \in N, \forall j \in F_i
\]  

(5)

\[
x_{ijk} \in \{0, 1\}, \quad y_{ij} \in \{0, 1\}, \quad \alpha_{ij} \in [0, 1]
\]
Optimization model

Server capacity constraints

\[
\text{Minimize} \quad \sum_{i \in N} \sum_{j \in F_i} \alpha_{ij} \cdot p_{bij} + (y_{ij} - \alpha_{ij}) \cdot p_{ij} \\
\text{Subject to} \quad \sum_{k \in M} d_k \cdot x_{ijk} \leq cap_{ij} \cdot y_{ij} \quad \forall i \in N, \forall j \in F_i \\
\sum_{i \in N} \sum_{j \in F_i} x_{ijk} = 1 \quad \forall k \in M \\
\sum_{j \in F_i} y_{ij} \leq 1 \quad \forall i \in N \\
\alpha_{ij} \leq y_{ij} \quad \forall i \in N, \forall j \in F_i \\
x_{ijk} \in \{0, 1\}, \quad y_{ij} \in \{0, 1\}, \quad \alpha_{ij} \in [0, 1]
\]
Optimization model

Application allocation constraints

Minimize \[ \sum_{i \in N} \sum_{j \in F_i} \alpha_{ij} \cdot p_{bij} + (y_{ij} - \alpha_{ij}) \cdot p_{ij} \] (1)

Subject to \[ \sum_{k \in M} d_k \cdot x_{ijk} \leq cap_{ij} \cdot y_{ij} \quad \forall i \in N, \forall j \in F_i \] (2)

\[ \sum_{i \in N} \sum_{j \in F_i} x_{ijk} = 1 \quad \forall k \in M \] (3)

\[ \sum_{j \in F_i} y_{ij} \leq 1 \quad \forall i \in N \] (4)

\[ \alpha_{ij} \leq y_{ij} \quad \forall i \in N, \forall j \in F_i \] (5)

\[ x_{ijk} \in \{0, 1\}, \ y_{ij} \in \{0, 1\}, \ \alpha_{ij} \in [0, 1] \]
Optimization model

Server frequency selection constraints

\[
\begin{align*}
\text{Minimize} & \quad \sum_{i \in N} \sum_{j \in F_i} \alpha_{ij} \cdot p_{bij} + (y_{ij} - \alpha_{ij}) \cdot p_{iij} \\
\text{Subject to} & \quad \sum_{k \in M} d_k \cdot x_{ijk} \leq cap_{ij} \cdot y_{ij} \quad \forall i \in N, \forall j \in F_i \\
& \quad \sum_{i \in N} \sum_{j \in F_i} x_{ijk} = 1 \quad \forall k \in M \\
& \quad \sum_{j \in F_i} y_{ij} \leq 1 \quad \forall i \in N \\
& \quad \alpha_{ij} \leq y_{ij} \quad \forall i \in N, \forall j \in F_i \\
& \quad x_{ijk} \in \{0, 1\}, \ y_{ij} \in \{0, 1\}, \ \alpha_{ij} \in [0, 1]
\end{align*}
\]
Optimization model

Bound constraints for utilization and server selection variables

\[
\begin{align*}
\text{Minimize} & \quad \sum_{i \in N} \sum_{j \in F_i} \alpha_{ij} \cdot p_{b_{ij}} + (y_{ij} - \alpha_{ij}) \cdot p_{i_{ij}} \\
\text{Subject to} & \quad \sum_{k \in M} d_k \cdot x_{ijk} \leq \text{cap}_{ij} \cdot y_{ij} \quad \forall i \in N, \, \forall j \in F_i \quad (2) \\
& \quad \sum_{i \in N} \sum_{j \in F_i} x_{ijk} = 1 \quad \forall k \in M \quad (3) \\
& \quad \sum_{j \in F_i} y_{ij} \leq 1 \quad \forall i \in N \quad (4) \\
& \quad \alpha_{ij} \leq y_{ij} \quad \forall i \in N, \, \forall j \in F_i \quad (5)
\end{align*}
\]

\[x_{ijk} \in \{0, 1\}, \quad y_{ij} \in \{0, 1\}, \quad \alpha_{ij} \in [0, 1]\]
Optimization model

Domains of the decision variables

\[
\begin{align*}
\text{Minimize} & \quad \sum_{i \in N} \sum_{j \in F_i} \alpha_{ij} \cdot pb_{ij} + (y_{ij} - \alpha_{ij}) \cdot p_{ij} \\
\text{Subject to} & \quad \sum_{k \in M} d_k \cdot x_{ijk} \leq cap_{ij} \cdot y_{ij} \quad \forall i \in N, \forall j \in F_i \\
& \quad \sum_{i \in N} \sum_{j \in F_i} x_{ijk} = 1 \quad \forall k \in M \\
& \quad \sum_{j \in F_i} y_{ij} \leq 1 \quad \forall i \in N \\
& \quad \alpha_{ij} \leq y_{ij} \quad \forall i \in N, \forall j \in F_i \\
\end{align*}
\]

\[x_{ijk} \in \{0, 1\}, \quad y_{ij} \in \{0, 1\}, \quad \alpha_{ij} \in [0, 1]\]
Extensions to the model

• Application workload balancing
  – Represent fractions of an application workload in a given selected server
  – Relax the allocation variable (to be a real domain)

• Server switching on/off and VM migration costs
  – Keep state from previous configuration
  – Add penalty to the objective function
Optimization control loop

do periodically:
   // 1. Input variables
   d = getDemandVector()
   curUsage = getCurrentUsage()
   curAlloc = getCurrentAlloc()

   // 2. Run optimization
   newUsage, newAlloc = bestConfig(d)

   // 3. Generate usage and alloc sets for changes
   chgUsage = sort(diff(newUsage, curUsage))
   chgAlloc = sort(diff(newAlloc, curAlloc))

   // 4. Power management operations
   for (i, j) in chgUsage:
      if j == 0:
         turnOff(i)
      else:
         if curUsage[i] == 0:
            turnOn(i)
         setFreq(i, j)

   // 5. Virtualization management operations
   for (k, i) in chgAlloc:
      if i == 0:
         stopVm(k, curAlloc[k])
      else:
         if curAlloc[k] == 0:
            startVm(k, i)
         else:
            migrateVm(k, curAlloc[k], i)
Optimization control loop

Control loop steps:
1. **Collect** and store the most recent values of the optimization input variables;
2. Construct and **solve** a new optimization problem instance, yielding a new optimal configuration;
3. **Apply** the changes in the system, transitioning the system to a new state given by the new optimized configuration.
do periodically:
   // 1. Input variables
   d = getDemandVector()
   curUsage = getCurrentUsage()
   curAlloc = getcurrentAlloc()

   // 2. Run optimization
   newUsage, newAlloc = bestConfig(d)

   // 3. Generate usage and alloc sets for changes
   chgUsage = sort(diff(newUsage, curUsage))
   chgAlloc = sort(diff(newAlloc, curAlloc))

   // 4. Power management operations
   for (i, j) in chgUsage:
      if j == 0:
         turnOff(i)
      else:
         if curUsage[i] == 0:
            turnOn(i)
         setFreq(i, j)

   // 5. Virtualization management operations
   for (k, i) in chgAlloc:
      if i == 0:
         stopVm(k, curAlloc[k])
      else:
         if curAlloc[k] == 0:
            startVm(k, i)
         else:
            migrateVm(k, curAlloc[k], i)


Control loop steps:
(1) **Collect** and store the most recent values of the optimization input variables;
(2) Construct and **solve** a new optimization problem instance, yielding a new optimal configuration;
(3) **Apply** the changes in the system, transitioning the system to a new state given by the new optimized configuration.
Optimization control loop

Control loop steps:
(1) **Collect** and store the most recent values of the optimization input variables;
(2) Construct and **solve** a new optimization problem instance, yielding a new optimal configuration;
(3) **Apply** the changes in the system, transitioning the system to a new state given by the new optimized configuration.
Configuration support

The optimization proposal relies on monitoring and configuration capabilities, such as VM migration and server on/off, which are described by means of an API [Petrucci et. al ACM SAC’09]
Server cluster architecture

Workload:
- Client 1
- Client 2
- Client 3
- Client 4
- Client 5
- Client 6

Applications:
- App 1
- App 2
- App 3

Dispatcher Server
- Optimization Framework
- Monitor
- Control

Servers:
- Server 1
  - VM1 App1
  - VM1 App2
  - VM1 App3
- Server 2
  - OFF
- Server 3
  - OFF
Server cluster architecture

Workload

Client 1
Client 2
Client 3
Client 4
Client 5
Client 6
Client 7
Client 8

Optimization Framework

Dispatcher Server

control

monitor

App 1

App 2

App 3

Server 1
VM1 App1
VM1 App2
VM1 App3

Server 2
VM2 App3

Server 3
OFF
Evaluation

• Three distinct app workloads based on WC98
• Cluster setup with 5 physical servers
• Optimization model implemented using the CPLEX 11 package solver
• Power/performance benchmark for the servers
  – Workload generator: httperf tool
  – Power monitor: LabView with USB DAQ
Workload traces

Figure 1: Workload traces for three different applications using HTTP logs from WC98
Figure 2: Dynamic optimization execution
Power consumption

Figure 3: Cluster power consumption
Power consumption

- Comparison with Linux kernel CPU governors
- Energy consumption estimation

\[ E = \sum_t \sum_{i,j} \alpha_{ij}^t \cdot pb_{ij} + (1 - \alpha_{ij}^t) \cdot pi_{ij} \]

\( i \) is an active server (and \( j \) is its operating frequency)
\( \alpha \) is utilization at time \( t \) (\( pb \) and \( pi \) are idle and busy power)

Performance governor \( \rightarrow 847,778.82 \text{J} = 235.49 \text{Wh} \)
On-demand governor \( \rightarrow 735,630.05 \text{J} = 204.34 \text{Wh} \)
Optimization approach \( \rightarrow 452,050.15 \text{J} = 125.57 \text{Wh} \)

47% compared to performance
38% compared to ondemand
Scalability simulation

- CPLEX with time limit of 180 seconds (related to a given control period)

<table>
<thead>
<tr>
<th>Server-App</th>
<th>Avg. (s)</th>
<th>Stdev. (s)</th>
<th>Max. (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(5,3)</td>
<td>0.022</td>
<td>0.018</td>
<td>0.070</td>
</tr>
<tr>
<td>(10,6)</td>
<td>0.054</td>
<td>0.035</td>
<td>0.250</td>
</tr>
<tr>
<td>(15,9)</td>
<td>0.062</td>
<td>0.038</td>
<td>0.240</td>
</tr>
<tr>
<td>(30,18)</td>
<td>0.392</td>
<td>0.913</td>
<td>8.610</td>
</tr>
<tr>
<td>(50,30)</td>
<td>13.630</td>
<td>29.959</td>
<td>180.010</td>
</tr>
<tr>
<td>(80,48)</td>
<td>58.941</td>
<td>53.570</td>
<td>180.020</td>
</tr>
<tr>
<td>(100,60)</td>
<td>80.135</td>
<td>52.394</td>
<td>180.030</td>
</tr>
</tbody>
</table>

- 180 executions (1800s of workload duration spaced by 10s) – one execution for each second
Scalability simulation

- Now including an optimality tolerance of 5% (180 executions)

<table>
<thead>
<tr>
<th>Server-App</th>
<th>Avg. (s)</th>
<th>Stdev. (s)</th>
<th>Max. (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(5,3)</td>
<td>0.006</td>
<td>0.008</td>
<td>0.040</td>
</tr>
<tr>
<td>(10,6)</td>
<td>0.023</td>
<td>0.022</td>
<td>0.100</td>
</tr>
<tr>
<td>(15,9)</td>
<td>0.031</td>
<td>0.030</td>
<td>0.130</td>
</tr>
<tr>
<td>(30,18)</td>
<td>0.062</td>
<td>0.067</td>
<td>0.540</td>
</tr>
<tr>
<td>(50,30)</td>
<td>0.139</td>
<td>0.281</td>
<td>2.390</td>
</tr>
<tr>
<td>(80,48)</td>
<td>0.267</td>
<td>0.235</td>
<td>3.000</td>
</tr>
<tr>
<td>(100,60)</td>
<td>0.481</td>
<td>0.409</td>
<td>3.080</td>
</tr>
<tr>
<td>(200,120)</td>
<td>2.893</td>
<td>1.993</td>
<td>11.550</td>
</tr>
<tr>
<td>(350,210)</td>
<td>16.488</td>
<td>12.979</td>
<td>75.440</td>
</tr>
<tr>
<td>(500,300)</td>
<td>48.409</td>
<td>41.472</td>
<td>181.030</td>
</tr>
</tbody>
</table>
Conclusion

• We proposed an optimization solution for power and performance management
  – Targeted for virtualized server clusters
• The proposal includes an optimization model and a control loop strategy
• Simulations showed practicability and attractive power reductions compared to Linux governors
  – Mainly because of server on/off mechanisms
Current work

- Experimental evaluation in a real cluster test-bed
  - Optimization control loop implementation
  - Xen hypervisor and Apache web servers
- Analysis of overhead/cost in imposing dynamic cluster configurations
  - E.g., VM deployment/replication, live migration
- Improvements in the optimization decisions by leveraging predictive information about the workload
  - Well-known techniques for load forecasting
- Acceleration of the optimization process (branch-and-bound)
  - Problem-specific heuristics (upper bound) input for CPLEX
  - Valid inequalities to improve dual solution limits (lower bounds) of the MIP model
Thank you!

The contemporary Art museum in Niteroi, Rio de Janeiro
Variable-sized bin packing problem

Demand (req/s)

- 34
- 17
- 123

Capacity (req/s)

- 98
- 106
- 163
- 217

unused

n items

m bins / knapsacks