Respecting temporal constraints in virtualised services

{Tommaso Cucinotta, Gaetano F. Anastasi}¹
{Luca Abeni}²

¹ Scuola Superiore Sant’Anna, Pisa, Italy
² Università di Trento, Trento, Italy

Seattle, July 20 - RTSOAA 2009
Motivations (1/3)

Distributed computing spread

Reasons:
- high speed Internet connections
- mobile devices diffusion

Consequences:
- new software design paradigms
- new business models
Motivations (2/3)

**Service Oriented Architectures**
Promising approach for distributed applications:
- service composition
- service distribution
- service interoperability

**Virtualisation**
Exploited by Service Oriented Infrastructures:
- scaling costs
- security and dependability
- transparent support of applications
Motivations (3/3)

Virtualisation Issues
Multiple VMs on the same node:
- interferences
- uncontrollable behaviour of services

Undesirable effects
- QoS fluctuations
- SLA violations
Contributions

- Achieving *predictability* of temporal behaviour of virtualised components
- Applying mechanisms designed for hierarchical *real-time* systems for analysis purposes
- Proving through experimental results, *effectiveness* in guaranteeing temporal isolation of multiple VMs
Model (1/2)

Host
- $\{ VM^k : k = a, b, \ldots \}$
- global scheduler

Guest $VM^k$
- $\mathcal{T}^k = \{ \tau^k_i : i = 1, 2, \ldots \}$
- local scheduler
Real-time task $\tau$

Sequence of jobs $J_{i,j}^k$
- arrival time $r_{i,j}^k$
- execution time $c_{i,j}^k$
- absolute deadline $d_{i,j}^k$

Periodic task $\tau_i^k = (C_i^k, T_i^k)$
- $C_i^k = \max_j\{c_{i,j}^k\}$
- $d_{i,j}^k = r_{i,j+1}^k = r_{i,j}^k + T_i^k$
Problem

Temporal behaviour of the guest hardly predictable

CDF of $\rho$ for $\{\tau_1 = (30, 150), \tau_2 = (50, 200)\}$

Real hardware

Virtual Machine
Solution

Adopt an appropriate strategy of soft real-time scheduling

How can we schedule the various VMs?

How can we schedule tasks inside VMs?
Scheduling approaches
Fixed Priority (1/2)

FP inter-VM scheduling
- Each $VM^k$ has a priority $p^k$
- Each $\tau_i \in T^k$ has priority $p^k_i$
- Priority to $VM^a$ over $VM^b$ means $p^a < p^b$

Drawbacks
- A higher priority VM can stall the lower ones
- A deadline miss can happen even if VM not consume more than expected
**Scheduling anomalies**

- If $p^a < p^b$ then $p_i^a < p_j^b$ even if $T_i^a > T_j^b$
- Global priority differs from optimum RM
- Higher response times than RM

\[
\{\tau_1^a = (30, 150), \tau_2^a = (50, 200)\}
\]

*VM^a with VM^b*
Scheduling approaches
Reservation-Based (1/3)

Resource Reservations
- Reserve hardware resources to tasks
- Hard reservations guarantee execution for exactly $Q$ time units each $P$

RR inter-VM scheduling
- Each $VM^k$ is scheduled through a reservation $RSV^k = (Q^k, P^k)$
Scheduling approaches
Reservation-Based (2/3)

**Advantages**
- Guarantee temporal isolation
- Each VM does not interfere with the others

**Adopted approach**
- Hard CBS (Constant Bandwidth Server)
- Implemented in the AQuoSA scheduler
Scheduling approaches
Reservation-Based (3/3)

No deadline miss by scheduling \( VM^a \) with \( VM^b \)

\[
\{ \tau^a_1 = (30, 150), \tau^a_2 = (50, 200) \} \{ \tau^b_1 = (30, 120), \tau^b_2 = (40, 240) \}
\]

\( VM^a \) with FP

\( VM^a \) with \( RSV^a = (28, 50) \)
### SOA perspective

Necessity to provide accurate estimations of service response times:
- Meet business policies
- Respecting QoS guarantees

### Common SOA scenario

- Multiple web servers in different VMs
- VMs concurrently running (same host)
- Different workloads

Respecting temporal constraints in virtualised services
Test model:

- \( VM^a \) and \( VM^b \) running Apache2
- \( VM^a \) serving 10 concurrent clients
- \( VM^b \) serving 20 concurrent clients
- Two kind of services:
  - \( req1 \) rotation of a 1000x1000 image
  - \( req2 \) rotation of a 2000x2000 image
### Response times for req2

<table>
<thead>
<tr>
<th></th>
<th>Stand-alone VM&lt;sup&gt;a&lt;/sup&gt;</th>
<th>Stand-alone VM&lt;sup&gt;b&lt;/sup&gt;</th>
<th>Concurrently VM&lt;sup&gt;a&lt;/sup&gt;</th>
<th>Concurrently VM&lt;sup&gt;b&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>avg</strong></td>
<td>0.769</td>
<td>1.603</td>
<td>1.564</td>
<td>2.416</td>
</tr>
<tr>
<td><strong>max</strong></td>
<td>7.700</td>
<td>14.114</td>
<td>13.253</td>
<td>23.056</td>
</tr>
<tr>
<td><strong>std.dev</strong></td>
<td>1.547</td>
<td>2.754</td>
<td>2.434</td>
<td>4.409</td>
</tr>
</tbody>
</table>

- Each VM is affected by the interference of the other
- Fluctuations are large and frequent
- **Impossibility to control service response times**
Attaching reservations to VMs:
- $VM^a$ served by $RSV^a = (40, 100)$
- $VM^b$ served by $RSV^b = (50, 100)$

Response times for req2

<table>
<thead>
<tr>
<th></th>
<th>Reservation $VM^a$ (s)</th>
<th>Reservation $VM^b$ (s)</th>
<th>Concurrently $VM^a$ (s)</th>
<th>Concurrently $VM^b$ (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg</td>
<td>1.961</td>
<td>3.146</td>
<td>1.564</td>
<td>2.416</td>
</tr>
<tr>
<td>max</td>
<td>10.424</td>
<td>19.508</td>
<td>13.253</td>
<td>23.056</td>
</tr>
<tr>
<td>std.dev</td>
<td>1.967</td>
<td>3.237</td>
<td>2.434</td>
<td>4.409</td>
</tr>
</tbody>
</table>

- Maximum reduced, lower degree of uncertainty
- Still interferences inside each VM
Attaching reservation to requests and VMs:

- \( req_1 \) by \( RSV = (4, 100) \); \( req_2 \) by \( RSV = (5, 100) \)
- \( VM^a \) by \( RSV^a = (3, 10) \); \( VM^b \) by \( RSV^b = (6, 10) \)

Response times for \( req_2 \)

<table>
<thead>
<tr>
<th></th>
<th>Only VM</th>
<th>VM + Req</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( VM^b )</td>
<td>( VM^b )</td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(s)</td>
</tr>
<tr>
<td><strong>avg</strong></td>
<td>1.603</td>
<td>3.881</td>
</tr>
<tr>
<td><strong>max</strong></td>
<td>14.114</td>
<td>6.793</td>
</tr>
<tr>
<td><strong>std.dev</strong></td>
<td>2.754</td>
<td>0.977</td>
</tr>
</tbody>
</table>

- Flattened response times
Flexible policies can be easily applied
To give more importance to certain requests just change parameters of a RSV

Response times varying Q
Conclusions

- Real-time and virtualisation is possible
- SOI can benefit of this
- Service-level guarantees can be provided to virtualised components
- Hard resource reservations is an effective mechanism to do this
Anastasi

Respecting temporal constraints in virtualised services