Dynamic request management algorithms for Web-based services in Cloud computing

Riccardo Lancellotti
Mauro Andreolini
Claudia Canali
Michele Colajanni

University of Modena and Reggio Emilia
Request management for Cloud Computing

- **Cloud**: large architecture based on virtualization
- **On-demand scalability**
  - OK for slowly changing workloads
- **Problems for highly variable workloads**
  - Flash crowds
  - Slashdot effect
- → **issues in request management**
- **Dispatching**:
  - Coarse grained decisions
- **Redirection**:
  - Last defense line against overload
  - Operates at the server level, with fine grained decisions
Redirection algorithms

- **Redirection → two decisions to take:**
  1. Should request $r$ be processed locally or redirected?
  2. If $r$ is redirected, which is the best alternative server $s_b$

- **Existing solutions:**
  - Threshold-based algorithms
    → lack of adaptivity, oversimplified model
  - Analytical models (M/M/1, M/G/1)
    → oversimplified performance model (mean time), high computational cost (off-line)

- **Our proposal:** performance gain prediction algorithm that forecasts the expected performance in case of redirection
VM request redirection model

- Time shared Virtual CPU with monitoring facility and a local dispatcher (for redirection)
- Storage space shared among multiple VM (e.g., NAS)
- Redirection can occur between VMs sharing storage (and hosting the same apps)

Redirection:
- Migration of user sessions
- Trade-off: load sharing vs. migration overhead
- Can exploit load information about local and neighbor servers
Performance Gain Prediction algorithm

- **Redirection decisions take into account:**
  - Delay $d$ for redirection (migration overhead)
  - Characteristics of request $r$ (computational cost $O(r)$)
  - Load on server $sa$ at time $t$
  - Load on server $sb$ at time $t$

- **Predict response time $T(r, sa, t)$ and $T(r, sb, t)$**
  - Redirect if $T(r, sa, t) > T(r, sb, t)$
  - Where $T(r, sb, t)$ includes delay for redirection

- **Must predict expected response time $T(r, s, t)$**
Prediction of response time

• Exploit time shared model of CPU
  – Time shared processor with $Q$ processes
    $\rightarrow$ each process receives $1/Q$ of processor resources
  – Based on URL we can infer computational cost of request $r$ $\rightarrow$ estimation of service time $O_r$

• Prediction of response time
  – $T(r, sa, t) = O_r (Qsa(t) + 1)$
  – $T(r, sb, t) = O_r (Qsb(t) + 1) + d$

• Redirection condition becomes
  – Redirect $iif O_r (Qsa(t) - Qsb(t)) > d$
Coping with data variability

- High variability in the raw samples of $Q$
- Assumption: $Q$ not changing (too much) during request service

$\rightarrow$ Use of smoothing techniques

- Double Exponential Smoothing (DES)

$$Q_{s}'(t) = \gamma Q_{s}(t) + (1-\gamma)(Q_{s}(t-t)+b_{Q}(t-\Delta t))$$

where:

$$b_{Q}(t) = \alpha (Q_{s}(t)-Q_{s}(t-\Delta t)) + (1-\alpha)b_{Q}(t-\Delta t)$$
Alternative algorithms

- **Threshold-based**
  - Evaluation of CPU utilization
    - Redirects \( iif \ \rho_{sa} > Thr \)
    - \( Thr=0.7 \) (commonly used value)

- **High variability in the samples**
  - Use of smoothing techniques
  - Fair comparison with Performance Gain Prediction algorithm

- **Baseline comparison**
  - Local processing *(No redirection)*
Experimental setup

• **Discrete simulator based on Omnet++ framework**

• **Virtualized infrastructure:**
  – 50 server supporting the same Web-based application

• **Workload characteristics:**
  – Overload on 50% of the servers

• **Different migration delays:**
  – From 0.1 to 2 seconds
Performance evaluation

- For both scenarios predictive algorithm outperforms the alternatives. Performance gain:
  - Nearly 20% w.r.t. Threshold-based algorithm
  - Up to 60% w.r.t. No redirection (Local)
Amount of redirection

• **Threshold-based algorithm**
  - Large amount of redirection
  - Redirection decisions non adaptive to migration delay

• **Performance Gain Prediction algorithm**
  - Redirects only when needed
  - Takes into account migration delay

<table>
<thead>
<tr>
<th>Redirection overhead</th>
<th>Performance gain prediction</th>
<th>Threshold-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>d=0.1s</td>
<td>12%</td>
<td>67%</td>
</tr>
<tr>
<td>d=2 s</td>
<td>21%</td>
<td>67%</td>
</tr>
</tbody>
</table>
• Performance gain prediction algorithm redirects mainly the resources with high computational costs
  – Redirection only when we identify a significant performance gain
Conclusions

- **Proposal of redirection algorithms to face request surges in large data centers**
  - Exploits information on process queue length
  - Use of predictive techniques to quantify the performance gain from redirection

- **Comparison with threshold-based existing algorithms**
  - Response time → reduction close to 20% (90-percentile)
  - Number of redirected requests → reduction up to 5 times
  - Performance Gain Prediction algorithm redirects only the “right” resources
Dynamic request management algorithms for Web-based services in Cloud computing

Riccardo Lancellotti
Mauro Andreolini
Claudia Canali
Michele Colajanni

University of Modena and Reggio Emilia