QoS Guarantees for Network Bandwidth in Private Clouds

Gaetano F. Anastasi, Massimo Coppola*, Patrizio Dazzi, Marco Distefano

CNR-ISTI, Pisa

CloudForward 2016 – Madrid, 18/10/2016
• Cloud customers and providers rely on SLA
• Network service quality heavily influences a class of applications (interactive, near-real-time)
  – Network bottlenecks
  – Avalanche of missed deadlines
• Enforcing bandwidth QoS
  – Mutually safe QoS
  – Flexibility and dinamicity
  – Low overhead
• Reuse
  – Based on (Linux) open-source and reliable technology
• Enforce QoS specified by a network SLA when managing the interaction of multiple VMs
• Minimize the impact on the management and virtualization software stack
• Employ a soft-real time approach
  – no new deployment will be negotiated that will cause the overall set of constraints to become unsatisfiable
Our contribution

• Definition of suitable SLA parameters for bandwidth allocation
• Design & develop a management technique based on admission control to allocate and enforce network QoS
• Enforce the negotiated QoS via the Linux Traffic Control mechanism
• Perform an experimental evaluation of Linux Traffic Control
Background Technology

- Simple QoS architecture
  - Negotiation layer
  - Provisioning layer (enforces)
- SLA negotiated with customers via **WS-Agreement**
  - *Agreement Templates* proposed
  - The negotiation step performs *Admission Control*
    - Check that the proposal is compatible with current resource allocation
    - Reserves resources if the proposal is feasible
• **Provisioning layer**
  
  – Cloud manager (e.g. **OpenNebula**) receives service requests
  
  – General Purpose OS with Real-time features to enforce QoS guarantees
    
    • **Linux** was shown to be up to the task  [see (1) (2)]
    • Overhead is acceptable
    • Can cope with unpredictability of load when co-locating VMs
    • Approach suited to Type-2 Hypervisor (e.g. **KVM**)

CloudForward 2016 - Madrid 18/10/2016
SLA specification

• We defined two simple SLA terms that can be associated to a VM at negotiation
  – *MinBandwidth* — minimum guaranteed
  – *MaxBandwidth* — maximum allowed

• Unallocated host bw capacity can be assigned to clients and allow them to approach *MaxBandwidth*

```
<wsag:ServiceDescriptionTerm
    wsag:Name="server_parameters"
    wsag:ServiceName="httpd">
    <qos:ServerParams xmlns:qos="schemas.webqos">
        <qos:NetMinBandwidth
            unit="Mbps"> 19 </qos:NetMinBandwidth>
        <qos:NetMaxBandwidth
            unit="Mbps"> 19 </qos:NetMaxBandwidth>
    </qos:ServerParams>
</wsag:ServiceDescriptionTerm>
```
Admission control

- Provider’s negotiation layer takes into account
  - $B^{eth}$: Max HW network capacity
  - $U^{res}$: reserved network bandwidth utilization $\leq 1$
  - For each customer, negotiated values $U_{c}^{\text{min}}$ and $U_{c}^{\text{max}}$
  - $U_{c}$: bandwidth actually reserved to customer $c$
  - $U_{r}$: min bandwidth requested by new service

- Admission check: $\sum_{c} U_{c}^{\text{min}} + U_{r} < U^{res}$

- Distributes possibly spare bandwidth (up to $U^{res}$) among the customers
  - Current “fairness” criteria splits evenly unreserved bandwidth among the $n$ customers (see paper)
Linux Traffic Control (TC)

- Linux Kernel module for rearranging traffic flow and scheduling packet transmission
- Bandwidth enforcement is done in TC creating and manipulating a hierarchy of TC objects
  - Queueing disciplines (qdisc) a scheduler of packets, can contain classes has rate and ceil properties for traffic flow
    - we used the Hierarchy Token Bucket
    - Filter classifies packets into output queues
    - Separate leaves are created in the TC hierarchy of a host machine for all active VMs, plus one for unreserved bandwidth
Applying Linux TC to bandwidth

- The hierarchy encoding all bandwidth limits is dynamically modified via **user-space tools**
  - We chose to use the **tcng** tool
- If admission control check is fine
  - Take note that bandwidth is reserved
  - Reconfigure the TC hierarchy

```plaintext
htb () {
  class (rate 100Mbps, ceil 100Mbps {
    $rewww1=class(rate 19Mbps) {sfq;};
    $rewww2=class(rate 19Mbps) {sfq;};
    $rewww3=class(rate 19Mbps) {sfq;};
    $rewww4=class(rate 19Mbps) {sfq;};
    $rewww5=class(rate 19Mbps) {sfq;};
    $other=class(rate 5Mbps, ceil 5Mbps {sfq;}
  }
}
```
Experiment settings

• 100 Mb networking
• Service for rotating greyscale images with parametric resolution
  – Server is 64bit x86 CPU
  – Images are on a ram disk, no disk I/O
• Multiple Clients querying the service set up on separate HW via ab – Apache Benchmarking tool
  – 5 clients on separate TCP ports
## Experiments

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Traffic Control</th>
<th>Network Overload</th>
<th>Unreserved traffic handled</th>
<th>Varying rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>N</td>
<td>N</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>B</td>
<td>N</td>
<td>Y</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>D</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>E</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

CDF of transmission time for Exp. A (no TC, no overload) shows irregular transfer times.
Experiments B, C

- B adds a “vampire task” eating bandwidth by sending UDP packets (12.5 KB every 1ms)
- C keeps the vampire but enables TC
  - TC makes the transmission times even, system is more deterministic
  - We verified a slightly lower variance in Exp.C without the “vampire”
    - Good isolation among clients, but the eager task can still affect them a bit
    - Limitation of TC under maximum load, keeping $\text{U}^{\text{res}}$ at 0.95 improves the reliability
Experiments D, E

• Exp. D leaves out of the traffic shaping hierarchy the UDP traffic (no “other” class in the TC hierarchy)
  – No rate limiting on UDP: unleash the vampire again!

• The shaping still works fine for TCP traffic
  – Smooth, predictable behaviour of the service, low StdDev
  – Average transmission time grows proportionally to the actual utilization of the network

• Exp. E measured the accuracy of the HTB algorithm
  – Shaping traffic at various grain levels: 1 - 0.1 - 0.01 MB/s
  – Error up to 1.4% with respect to the expected bandwidth

CloudForward 2016 - Madrid 18/10/2016
Virtualized scenario (KVM)

- Server VM: 2 cores, 2GB RAM, 100Mbps NIC
  - HW: a 3.2GHz Xeon 16GB RAM
- Clients run on a 2.53GHz Xeon 112GB RAM
- TC is effective even when inside a VM to flatten the transmission times
- Overload scenario: TC reduces the risk of failing to provide the stipulated QoS
Conclusions

• Experimental evaluation of TC for network SLA enforcing
  – 0.95 reserved bandwidth capacity can be managed with high stability
  – Unallocated traffic shall be explicitly accounted for
    • Both TCP and UDP

• Shaping improves the regularity of the system, making it mode deterministic
  – Usable granularity of shaping is well down to 10Kb/s
  – Experiments confirm results in a virtualized scenario
Future Work

• Simulate and test larger VM networks, more complex and real test cases
• Investigate on the impact on application metrics for different applications
  – E.g. other “fairness” criteria for distributing unreserved bandwidth
• Test the approach in the Basmati EU-KR Project
  – Smart resource management in Federated Clouds (AI, ML and model-based)
Acknowledgments

• Thanks for listening!

• We acknowledge support of the BASMATI EU-KR joint research project

• Our experiments are based on previous work from RETIS Lab, SSSUP S.Anna, Pisa


• We acknowledge support of previous EU Projects
  • FP7-257438 Contrail
    Open Computing Infrastructures for Elastic Services
  • FP7-256980 NESSoS
    Network of Excellence on Engineering Secure Future Internet Software Services and Systems

CloudForward 2016 - Madrid 18/10/2016
Thanks!
Table 2. Service transmission times (ms) for Experiments A-D

<table>
<thead>
<tr>
<th>Experiment</th>
<th>min</th>
<th>avg</th>
<th>max</th>
<th>std.dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment A</td>
<td>38</td>
<td>117.95</td>
<td>158</td>
<td>36.63</td>
</tr>
<tr>
<td>Experiment B</td>
<td>124</td>
<td>195.3</td>
<td>226</td>
<td>26.28</td>
</tr>
<tr>
<td>Experiment C</td>
<td>186</td>
<td>191.3</td>
<td>193</td>
<td>1.51</td>
</tr>
<tr>
<td>Experiment D</td>
<td>222</td>
<td>230.6</td>
<td>237</td>
<td>3.03</td>
</tr>
</tbody>
</table>