Opportunistic Flooding to Improve TCP Transmit Performance in Virtualized Clouds

Presenter: Lucian Cojocar

February 29, 2012
1. Introduction
   - The problem

2. Solution
   - Intuition

3. Solution design
   - Congestion Control Module

4. Evaluation
   - TCP Throughput Evaluation
   - Real-life test

5. Conclusion
Introduction

The problem

Solution

Intuition

Solution design

Congestion Control Module

Evaluation

TCP Throughput Evaluation

Real-life test

Conclusion
Context

- VM consolidation
Context

- VM consolidation
- Multi-core systems
- *rapid* increase in VM density
- 25% – 11-25 VMs per machine
- 12% – 25 VMs per machine
Context

- VM consolidation
- Multi-core systems
- *rapid* increase in VM density
- 25% – 11-25 VMs per machine
- 12% – 25 VMs per machine
- same data center
Introduction

The problem

Problem

- high bandwidth rates (Gbps)
- low latency (< 1 ms)
- big timeslices (30 ms)
Problem

- high bandwidth rates (Gbps)
- low latency (< 1 ms)
- big timeslices (30 ms)

*Negative impact on TCP performance*
Table of Contents

1 Introduction
   • The problem

2 Solution
   • Intuition

3 Solution design
   • Congestion Control Module

4 Evaluation
   • TCP Throughput Evaluation
   • Real-life test

5 Conclusion
Paravirtualization
Key Intuition

Scheduled VM

Shared Buffer

Driver Domain

TCP Receiver

VM1

Data

Data

Data

Time

VM2

ACK

ACK

Vanilla VMM

VM3

ACK

Data

VM1

Data
Key Intuition (cont.)

- the driver domain (DD) is scheduled often
- the applications pushes data through DD
Key Intuition (cont.)

- the driver domain (DD) is scheduled often
- the applications pushes data through DD
- DD can push more data segments
- on behalf of VM, DD emulates TCP connection states
Table of Contents

1 Introduction
   • The problem

2 Solution
   • Intuition

3 Solution design
   • Congestion Control Module

4 Evaluation
   • TCP Throughput Evaluation
   • Real-life test

5 Conclusion
vFlood

With vFlood
- highest sequence number ACKed
- count of unACKed
- advertised recv. window
- window scaling factor
- buffer usage
Table of Contents

1 Introduction
   • The problem

2 Solution
   • Intuition

3 Solution design
   • Congestion Control Module

4 Evaluation
   • TCP Throughput Evaluation
   • Real-life test

5 Conclusion
Different flow size

Figure: vFlood improvement for different flow sizes
Scalable?

**Figure:** Concurrent flows
Apache Olio Benchmark
Apache Olio Benchmark
Table of Contents

1. Introduction
   - The problem

2. Solution
   - Intuition

3. Solution design
   - Congestion Control Module

4. Evaluation
   - TCP Throughput Evaluation
   - Real-life test

5. Conclusion
• *increasing* density of VMs per physical machine
- increasing density of VMs per physical machine
- TCP performance degrades
- increasing density of VMs per physical machine
- TCP performance degrades
- opportunistic flooding
Different loads

Figure: 3 VMs, 60% load
vFlood Overhead

<table>
<thead>
<tr>
<th>vFlood Routine</th>
<th>CPU Cycles</th>
<th>CPU %</th>
</tr>
</thead>
<tbody>
<tr>
<td>vFlood_tx()</td>
<td>65</td>
<td>0.62</td>
</tr>
<tr>
<td>vFlood_rx()</td>
<td>370</td>
<td>3.05</td>
</tr>
<tr>
<td>vFlood_hash_lookup()</td>
<td>78</td>
<td>0.73</td>
</tr>
<tr>
<td>vFlood_update_VM()</td>
<td>59</td>
<td>0.56</td>
</tr>
<tr>
<td>vFlood_process_threshold()</td>
<td>57</td>
<td>0.92</td>
</tr>
</tbody>
</table>

**Figure:** vFlood per-packet CPU usage
Apache Olio benchmark

<table>
<thead>
<tr>
<th>Operation</th>
<th>Count Vanilla Xen</th>
<th>Count vFlood</th>
<th>Count vSnoop</th>
<th>Count vFlood + vSnoop</th>
</tr>
</thead>
<tbody>
<tr>
<td>HomePage</td>
<td>2544</td>
<td>3271</td>
<td>3416</td>
<td>4215</td>
</tr>
<tr>
<td>TagSearch</td>
<td>3290</td>
<td>4281</td>
<td>4020</td>
<td>5550</td>
</tr>
<tr>
<td>EventDetail</td>
<td>2363</td>
<td>3077</td>
<td>3135</td>
<td>3925</td>
</tr>
<tr>
<td>PersonDetail</td>
<td>219</td>
<td>331</td>
<td>312</td>
<td>410</td>
</tr>
<tr>
<td>AddPerson</td>
<td>53</td>
<td>96</td>
<td>71</td>
<td>123</td>
</tr>
<tr>
<td>AddEvent</td>
<td>156</td>
<td>245</td>
<td>178</td>
<td>257</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>9512</strong></td>
<td><strong>12642</strong></td>
<td><strong>11940</strong></td>
<td><strong>15167</strong></td>
</tr>
<tr>
<td><strong>Rate(ops/sec)</strong></td>
<td><strong>31.7</strong></td>
<td><strong>42.1</strong></td>
<td><strong>39.8</strong></td>
<td><strong>50.5</strong></td>
</tr>
<tr>
<td><strong>Percentage Improvement</strong></td>
<td>-</td>
<td><strong>32.9%</strong></td>
<td><strong>25.5%</strong></td>
<td><strong>59.5%</strong></td>
</tr>
</tbody>
</table>

**Figure:** Apache Olio full benchmark (with vSnoop)
Buffer management policies

Figure: Buffer management policies
vFlood state machine

Figure: vFlood state machine
**Figure:** vFlood architecture
vFlood implementation

Figure: vFlood implementation
# Apache Olio Benchmark

<table>
<thead>
<tr>
<th>Operation</th>
<th>Vanilla Xen</th>
<th>vFlood</th>
</tr>
</thead>
<tbody>
<tr>
<td>HomePage</td>
<td>2544</td>
<td>3271</td>
</tr>
<tr>
<td>TagSearch</td>
<td>3290</td>
<td>4281</td>
</tr>
<tr>
<td>EventDetail</td>
<td>2363</td>
<td>3077</td>
</tr>
<tr>
<td>PersonDetail</td>
<td>219</td>
<td>331</td>
</tr>
<tr>
<td>AddPerson</td>
<td>53</td>
<td>96</td>
</tr>
<tr>
<td>AddEvent</td>
<td>156</td>
<td>245</td>
</tr>
<tr>
<td>Rate(ops/sec)</td>
<td>31.7</td>
<td>42.1</td>
</tr>
<tr>
<td>Percentage improvement</td>
<td>–</td>
<td>32.9%</td>
</tr>
</tbody>
</table>