EyeQ
Protecting Network Performance in the Cloud

Vimal

Mohammad Alizadeh
Balaji Prabhakar
David Mazières

Changhoon Kim
Albert Greenberg

Windows Azure
Once upon a time...
Once upon a time...
Once upon a time...
Once upon a time...
Performance Unpredictability

Graph (Sun Apr 14 12:43:20 EDT 2013 to Mon Apr 15 14:00:00 EDT 2013):

150ms

99th pcile

Mean, Median

http://amistrongeryet.com/op_detail.jsp?
op=gae_db_readCachedHandles_1&hoursAgo=24
99th percentile latency: Who cares?

Web services: each request touches 10s to 100s of servers

Web Response time depends on the slowest worker.
As N increases, 99th percentile latency really matters
Network Congestion Kills Predictability
Is this how we deal with variability?
Is this how we deal with variability?

Why We Moved Off The Cloud

The cloud’s intractable problem

... is variable — no, highly variable — performance.


Give up on cloud, move to dedicated
Is this how we deal with variability?

Why We Moved Off The Cloud

The cloud’s intractable problem

... is variable — no, highly variable — performance.


5 Lessons We’ve Learned Using AWS

... in the Netflix data centers, we have a high capacity, super fast, highly reliable network. This has afforded us the luxury of designing around chatty APIs to remote systems. AWS networking has more variable latency.


Give up on cloud, move to dedicated

Overhaul apps to deal with variability
Congestion is notorious you when you can’t “see” it

1 Long lived TCP flow

Bursty UDP session:
2.5Gb/s
ON: 5ms
OFF: 15ms
Congestion is notorious when you can’t “see” it

1 Long lived TCP flow

Switch

Shared 10G pipe

Bursty UDP session:
**2.5Gb/s**
ON: 5ms
OFF: 15ms
Congestion is notorious you when you can’t “see” it

1 Long lived TCP flow

Switch

Shared 10G pipe

Bursty UDP session:
2.5Gb/s
ON: 5ms
OFF: 15ms
Strawman: Rate Limiting

Rate Limit: 2.5Gb/s

Shared Network Capacity (10Gb/s)

Rate Limit: 7.5Gb/s
Strawman: Rate Limiting

- Rate Limit: 2.5Gb/s
- Shared Network Capacity (10Gb/s)
- Rate Limit: 7.5Gb/s
Strawman: Rate Limiting

Rate Limit: 2.5Gb/s

Shared Network Capacity (10Gb/s)

Rate Limit: 7.5Gb/s
Recap...

- **Bandwidth contention**
  - Can occur in a few milliseconds!
  - Happens even if tenants don’t talk to each other
  - Invisible on 5 minute logs!
- **Cannot trust tenants at all**
  - Easy to grab more bandwidth simply by blasting traffic (UDP)
- **Naïve rate limiting not enough**
  - VMs can gang up and blast traffic
Where does Congestion Happen?

Ideal network fabric (one switch)

- Tenant 1
- Tenant 2

Shared 10Gb/s pipe

Server

VM

VM

VM

VM
Where does Congestion Happen?

Ideal network fabric (one switch)

Shared 10Gb/s pipe

Server

VM

VM

VM

VM

Tenant 1

Tenant 2
Where does Congestion Happen?

Ideal network fabric (one switch)

Tenant 1

Tenant 2

15 Apr 2013
Where does Congestion Happen?

Tenant 1

Tenant 2

VM

VM

VM

VM

Shared 10Gb/s pipe

Server

Ideal network fabric (one switch)
Where does Congestion Happen?

Ideal network fabric (one switch)
Congestion Study on Windows Azure

Spine Layer

Leaf layer

160Gb/s
Leaf to Spine

480Gb/s
in a Rack

< 3:1
oversub

15 Apr 2013
Hottest cluster: 1000x more drops at the Edge, than Core.

16 of 17 clusters: 0 drops in the Core.

Timescales: over 2 weeks, 99.9th percentile = several minutes
EyeQ’s Goal: Rate Guarantees for VMs in the Cloud

**Customer** specifies capacity of the virtual NIC.

**Provider**: assures near dedicated performance.

Rate guarantees => Performance isolation
The Big Picture: Resource Management

Apache Mesos
At Twitter, AirBnB, Conviva, etc.

Fair CPU Scheduling

EyeQ
Network Rate Guarantees
Decentralized Scheduling

EyeQ Shim Layer In the Trusted Domain (Hypervisor/NIC)
Decentralized Scheduling
Decentralized Scheduling
Decentralized Scheduling

2Gb/s → VM

1Gb/s

2Gb/s → VM

1Gb/s

8Gb/s → VM

8Gb/s

8Gb/s → VM

8Gb/s

15 Apr 2013
Decentralized Scheduling
Work Conserving Allocations

RX Module

2Gb/s → VM

1Gb/s → VM

1Gb/s → VM

5Gb/s → VM

8Gb/s → VM

5Gb/s → VM
Work Conserving Allocations

RX Module

Spare capacity

2Gb/s VM

1Gb/s

1Gb/s

5Gb/s

5Gb/s

8Gb/s VM

2Gb/s VM

8Gb/s VM

8Gb/s VM

15 Apr 2013
Work Conserving Allocations
Transmit/Receive Modules

Per-destination rate limiters:
only if dest. is congested... bypass otherwise
Transmit/Receive Modules

RCP: Rate feedback (R) every 10kB (no per-source state needed)

Per-destination rate limiters:
only if dest. is congested... bypass otherwise
**Transmit/Receive Modules**

RCP: Rate feedback (R) every 10kB (no per-source state needed)

Per-destination rate limiters:
only if dest. is congested... bypass otherwise
Transmit/Receive Modules

RCP: Rate feedback (R) every 10kB (no per-source state needed)

Feedback pkt Rate: 1Gb/s

Congestion detectors

Per-destination rate limiters:
only if dest. is congested... bypass otherwise
Determining Rate

Determine one rate $R_i$ so that utilisation matches allowed limits.

Recompute $R_i$ every 200us

$$R_i \leftarrow R_i \left( 1 - \alpha \cdot \frac{y_i}{C_i} \right)$$

Aggressiveness parameter. Set to 0.5
Determining Rate

Start at line rate:
10Gb/s

3Gb/s (VM’s limit)
Determining Rate (distributed guessing)

Too high: try 3Gb/s

3Gb/s

VM

VM

VM

Tiny feedback packets sent to traffic sources

15 Apr 2013
More guessing...

Net Rate=9Gb/s
Continue guessing...

Too high: try 0.5Gb/s
Oops…

Net Rate = 1.5 Gb/s

3 Gb/s
Almost there

Too low: try 1Gb/s

15 Apr 2013
Fixed point

Net Rate=3Gb/s
Continuous Scheduling

Net Rate = 2 Gb/s

3 Gb/s

VM

VM

15 Apr 2013
Continuous Scheduling

This happens every 200us.

Too low: try 1.5Gb/s
Software Prototype

Linux Kernel Module (qdisc)  
Windows Filter Driver (in VMSwitch)

• Non-intrusive: no changes to applications or existing network stack. Works even with UDP!
• ~1700 lines of code

Linux Kernel Module is Open-Source

• Full system and documentation at http://jvimal.github.com/eyeq
• EyeQ’s rate limiters more efficient than today’s rate limiters in Linux/Windows
Does it work?

TCP: 6Gb/s
UDP: 3Gb/s
Does it work?

TCP: 6Gb/s
UDP: 3Gb/s

Without EyeQ
Does it work?

TCP: 6Gb/s
UDP: 3Gb/s

Without EyeQ

With EyeQ
Does it work?

TCP: 6Gb/s
UDP: 3Gb/s

Improves utilisation

Provides protection

Without EyeQ
With EyeQ
Does it work?

TCP: 6Gb/s
UDP: 3Gb/s

**Improves utilisation**

**Provides protection**

Without EyeQ

With EyeQ
Close to Bare-Metal Latency?

Each server has 10Gb/s link

12 Client Pool

4 Server Pool
Close to Bare-Metal Latency?

Each server has 10Gb/s link

External Load: 144k SET req/sec

12 Client Pool

Set 6kB objects
Load: 2.3Gb/s/server

4 Server Pool
Close to Bare-Metal Latency?

Each server has 10Gb/s link

External Load: 144k SET req/sec

Set 6kB objects
Load: 2.3Gb/s/server

12 Client Pool

4 Server Pool
Close to Bare-Metal Latency?

Each server has 10Gb/s link

External Load: 144k SET req/sec

Set 6kB objects
Load: 2.3Gb/s/server

12 Client Pool

4 Server Pool
Close to Bare-Metal Latency?

12 Client Pool

Each server has 10Gb/s link

Set 6kB objects
Load: 2.3Gb/s/server

UDP bursty 5Gb/s
0.5s to 1 server, chosen round robin. 0.5s sleep between bursts.

4 Server Pool

External Load: 144k SET req/sec

Load: 2.3Gb/s/server

15 Apr 2013
Close to Bare-Metal Latency?

Each server has 10Gb/s link

External Load: 144k SET req/sec

Set 6kB objects
Load: 2.3Gb/s/server

UDP bursty 5Gb/s
0.5s to 1 server, chosen round robin. 0.5s sleep between bursts.

12 Client Pool

4 Server Pool
Close to Bare-Metal Latency?

Each server has 10Gb/s link

External Load: 144k SET req/sec

Set 6kB objects
Load: 2.3Gb/s/server

UDP bursty 5Gb/s
0.5s to 1 server, chosen round robin. 0.5s sleep between bursts.

12 Client Pool

4 Server Pool
Close to Bare-Metal Latency?

Set 6kB objects
Load: 2.3Gb/s/server

Each server has
10Gb/s link

<table>
<thead>
<tr>
<th>Scenario</th>
<th>50\textsuperscript{th}</th>
<th>99.9\textsuperscript{th}</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (Linux 3.4)</td>
<td>98us</td>
<td>666us</td>
<td>144kreq/s</td>
</tr>
<tr>
<td>Without Interference + EyeQ</td>
<td>100us</td>
<td>630us</td>
<td>144kreq/s</td>
</tr>
<tr>
<td>With Interference</td>
<td>4127us</td>
<td>&gt;10^6us</td>
<td>144kreq/s</td>
</tr>
<tr>
<td>With Interference + EyeQ</td>
<td>102us</td>
<td>750us</td>
<td>144kreq/s</td>
</tr>
</tbody>
</table>

12 Client Pool   4 Server Pool

15 Apr 2013
Thank you!

EyeQ: a system to partition bandwidth within a data center in a simple and predictable way

http://jvimal.github.com/eyeq
jvimal@stanford.edu
Rate Limiter Memory Overhead

112B + NCUPS * 104B

• 8 CPUs: ~0.9kB
• 16 CPUs: ~1.8kB
• Scales linearly with number of IP destinations, not connections (struct sock: 648B)