Cloud Scale Load Balancing

- Ananta: Cloud scale load balancing, Parveen Patel et.al; SIGCOMM 2013
- Duet: Cloud scale Load Balancing with hardware and software, Rohan Gandhi, et.al; SIGCOMM 2014

Slides taken from SIGCOMM presentation
What’s happening?

• Cloud Services
• Azure (Microsoft), AWS (Amazon), Google Cloud Platform (Google)
• Multi-tenant
• Huge Compute Capacity
• Needs to be shared with high utilization
• Spread requests over compute/storage resources
• Load balancer in the front end
Windows Azure - Some Stats

• More than 50% of Fortune 500 companies using Azure

• Nearly 1000 customers signing up every day

• Hundreds of thousands of servers

• We are doubling compute and storage capacity every 6-9 months

• Azure Storage is Massive – over 4 trillion objects stored
Scaling

• Vertical scaling (scale up) vs Horizontal scaling (scale out)

• Vertical scaling
  – More resource/capacity to a node/switch/server
  – Increase power of resource, bigger
  – Reliability

• Horizontal scaling
  – More independent elements/nodes to a system
  – Increase resources, more of the same
  – Distributed state
  – failover
Load balancing

• Rotating DNS
• Configure DNS to return different IP addresses for the same name resolution
  • www.yahoo.com --- x.y.z.1, x.y.z.2 ....
• Advantages, Layer 3.. Scales
• Disadvantages: DNS does not know the load or the health of the servers
• Cannot use heterogeneous load balancing policy
• Most load balancers are switch based . Layer4-Layer7
Layer 4 LB/switching

• Switch is configured to have virtual IP address (VIP)
• This is the IP address that the DNS returns for the name
• Switch configured based on policy to determine which actual server the SYN is forwarded
• Can be based on destination port# (80 for http, 21 for SMTP, etc)
• Acts as a destination-NAT
• Return packets come to switch and the Source address changed back to VIP
Layer 4 LB

<table>
<thead>
<tr>
<th>VIP</th>
<th>VIP-P</th>
<th>DIP</th>
<th>DP</th>
</tr>
</thead>
<tbody>
<tr>
<td>141.1.2.6</td>
<td>80</td>
<td>10.1.2.3</td>
<td>80</td>
</tr>
<tr>
<td>141.1.2.6</td>
<td>21</td>
<td>10.1.2.4</td>
<td>21</td>
</tr>
</tbody>
</table>
Layer 4 LB

- Port aware
- Switching decisions based on TCP SYN
- Destination NAT
- Return packets need to come back to the switch/LB
- Need to act as Source NAT as well
- LB becomes bottleneck
Direct path to sender

• With a NAT box technique for rewriting, packets need to return to the switch
• Better to avoid going back to the LB on the return path
• Direct return to sender
• Need IP-in-IP encapsulation
Layer 4 vs layer 7 load balancing

- Layer 4 LB based on TCP header only
- Layer 7 LB examines the content and then decides the server.
- Based on http requests.
- Layer 7 switches need to terminate TCP connections
- LB needs to wait for http request before deciding on the binding
- External Client terminates TCP connection at switch
- After application level request, binding determined to application server.
- The client still need to see the same sequence numbers coming from the application server
- Need to imitate the same connection as the switch.
TCP Splicing

Client

`SYN(CSEQ)`

`HTTPREQ(CSEQ+1)`

`ACK(DSEQ+1)`

L7 Switch

Step 1

`SYN(DSEQ)`

Step 2

`ACK(CSEQ+1)`

Step 3

`HTTPREQ(CSEQ+1)`

Step 4

`ACK(DSEQ+1)`

Step 5

`DATA(SSEQ+1)`

Step 6

`ACK(CSEQ+HTTPREQ+1)`

Step 7

`ACK(SSEQ+HTTPRES+1)`

Server

`ACK(DSEQ + HTTPRES + 1)`

`ACK(SSEQ + HTTPRES+1)`

`DATA(SSEQ+1)`

`ACK(CSEQ+HTTPREQ+1)`

`ACK(SSEQ+ HTTPRES+1)`

`ACK(SSEQ+ HTTPRES+1)`

`DATA(SSEQ+1)`
Ananta in a nutshell

• Is NOT hardware load balancer code running on commodity hardware

• Is distributed, scalable architecture for Layer-4 load balancing and NAT

• Has been in production in Bing and Azure for three years serving multiple Tbps of traffic

• Key benefits
  – Scale on demand, higher reliability, lower cost, flexibility to innovate
How are load balancing and NAT used in Azure?
Background: Inbound VIP communication

Terminology:
VIP – Virtual IP
DIP – Direct IP

LB load balances and NATs VIP traffic to DIPs

VIP = 1.2.3.4

DIP = 10.0.1.1  DIP = 10.0.1.2  DIP = 10.0.1.3
Background: Outbound (SNAT) VIP communication

VIP1 = 1.2.3.4
DIP = 10.0.1.1
Front-end VM
DIP = 10.0.1.20
Back-end VM

VIP2 = 5.6.7.8
DIP = 10.0.2.1
Front-end VM
DIP = 10.0.2.2
Front-end VM
DIP = 10.0.2.3
Front-end VM

1.2.3.4 → 5.6.7.8
Service 1
Service 2
VIP traffic in a data center

**Total Traffic**
- VIP Traffic 44%
- DIP Traffic 56%

**VIP Traffic**
- Internet 14%
- Inter-DC 16%
- Intra-DC 70%

**VIP Traffic**
- Outbound 50%
- Inbound 50%
Why does our world need yet another load balancer?
Traditional LB/NAT design does not meet cloud requirements

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Details</th>
<th>State-of-the-art</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scale</td>
<td>• Throughput: ~40 Tbps using 400 servers</td>
<td>• 20Gbps for $80,000</td>
</tr>
<tr>
<td></td>
<td>• 100Gbps for a single VIP</td>
<td>• Up to 20Gbps per VIP</td>
</tr>
<tr>
<td></td>
<td>• Configure 1000s of VIPs in seconds in the event of a disaster</td>
<td>• One VIP/sec configuration rate</td>
</tr>
<tr>
<td>Reliability</td>
<td>• N+1 redundancy</td>
<td>• 1+1 redundancy or slow failover</td>
</tr>
<tr>
<td></td>
<td>• Quick failover</td>
<td></td>
</tr>
<tr>
<td>Any service anywhere</td>
<td>• Servers and LB/NAT are placed across L2 boundaries for scalability</td>
<td>• NAT and Direct Server Return (DSR) supported only in the same L2</td>
</tr>
<tr>
<td></td>
<td>and flexibility</td>
<td></td>
</tr>
<tr>
<td>Tenant isolation</td>
<td>• An overloaded or abusive tenant cannot affect other tenants</td>
<td>• Excessive SNAT from one tenant causes complete outage</td>
</tr>
</tbody>
</table>
Key idea: decompose and distribute functionality

VIP Configuration:
VIP, ports, # DIPs

Software router
(Needs to scale
to Internet bandwidth)

Hosts
(Scales naturally with
# of servers)
Ananta: data plane

1st Tier: Provides packet-level (layer-3) load spreading, implemented in routers via ECMP.

2nd Tier: Provides connection-level (layer-4) load spreading, implemented in servers.

3rd Tier: Provides stateful NAT implemented in the virtual switch in every server.
Inbound connections

Packet Headers

1. Dest: VIP, Src: Client
2. Dest: DIP, Src: MUX
3. Dest: VIP, Src: Client

Client

Router

MUX

Host Agent

Host

VM DIP

Packet Headers

Dest: ClientVIP

Microsoft
Outbound (SNAT) connections

- **Control Packets**
- **Data Packets**

1. **Host Agent**
2. **Ananta Manager**
3. **VIP:1025 → DIP2**
4. **Host**
5. **DIP2**
6. **Router**
7. **MUX**
8. **VM (DIP1)**

- **Packet Headers**
  - **Dest:** Server:80
  - **Src:** VIP:1025

- **Dest:** Server:80
- **Src:** DIP2:5555
Fastpath: forward traffic

VIP1

MUX 1

VIP2

MUX 2

Host

Host Agent

VM DIP1

Host

Host Agent

VM DIP2

Destination

Data Packets

SYN

SOURCE=VIP1

Microsoft
Fastpath: return traffic

VIP1 -> MUX 1 -> Host Agent
1. SOURCE=VIP2, SYN, DEST=VIP1
2. SYN-ACK

VIP2 -> MUX 2 -> Host Agent
3. DATA PAKETS
4. Host Agent -> Destination

VIP1
Host
DIP1
...

VIP2
Host
DIP2
...

MUX 1
...

MUX 2
...
Fastpath: redirect packets
Fastpath: low latency and high bandwidth for intra-DC traffic

- Redirect Packets
- Data Packets

VIP1
MUX 1

VIP2
MUX 2

Host
Agent
VM DIP1

8

Host
Agent
VM DIP2

Destination

Microsoft
Impact of Fastpath on Mux and Host

CPU

% CPU

No Fastpath

Fastpath

Host

Mux

Microsoft
Lessons learnt

• Centralized controllers work
  – There are significant challenges in doing per-flow processing, e.g., SNAT
  – Provide overall higher reliability and easier to manage system

• Co-location of control plane and data plane provides faster local recovery
  – Fate sharing eliminates the need for a separate, highly-available management channel

• Protocol semantics are violated on the Internet
  – Bugs in external code forced us to change network MTU

• Owning our own software has been a key enabler for:
  – Faster turn-around on bugs, DoS detection, flexibility to design new features
  – Better monitoring and management

Microsoft
Duet: Cloud scale LB with hardware and Software
LB architecture

Internet

Datacenter

VIP1, VIP2

Load balancer (Mux)

Store VIP-DIP mapping
Split VIP traffic to DIPs

Server
DIP = 10.0.1.1
Server
DIP = 10.0.2.2

bing.com
VIP1 = 1.1.1.1

Server
DIP = 10.0.3.3
Server
DIP = 10.0.4.4

OneDrive
VIP2 = 1.1.1.2

Terminology
VIP – Virtual IP
DIP – Direct IP
Mux – Multiplexer
Hardware vs software LB

- Software LB
- Horizontal Scaling
- Distributed deployment
- High availability
- Slow, latency penalty

- Hardware
- Fast, low latency
- Costly
- Vertical scaling
- Failover difficult
Limitations of SW LB

High latency inflation: 200 usec
Low capacity: 300k pkts/sec

5k SMuxes needed at 15Tbps traffic in 50k server DC
Basic idea of using HW LB

• Use Commodity switches as LB
• Use forwarding tables already available
• Use switch resources
• But
  – Limited table space
  – Partition VIP space (make many switches to work as Hardware LB)
• But
  – Need to route VIP traffic to these switches
• Deal with failures of switches
Implementing HMux on switch

- Make use of ECMP table and Tunneling
- Split traffic across DIPS
- Forward VIP traffic to DIPS
Challenges

• Limited S/W memory
• Availability
• VIP assignment
• VIP migration
Switches have limited memory

Workload: 64K VIPS or more; Millions of DIPs

<table>
<thead>
<tr>
<th>Table</th>
<th>Forwarding Table</th>
<th>ECMP</th>
<th>Tunnel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>16K</td>
<td>4K</td>
<td>512</td>
</tr>
</tbody>
</table>
Partition VIPs across HLBs

<table>
<thead>
<tr>
<th>VIP</th>
<th>DIPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.1.1/32</td>
<td>DIP1, DIP2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VIP</th>
<th>DIPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.1.2/32</td>
<td>DIP3, DIP4</td>
</tr>
</tbody>
</table>

Source

DIP2

DIP4
Availability?

Block of VIPs not accessible? Use Software backup
Use Software LB as a backup

<table>
<thead>
<tr>
<th>VIP</th>
<th>DIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.1.1/32</td>
<td>DIP1, DIP2</td>
</tr>
</tbody>
</table>

Software LBs can handle entire range of VIPs like Ananta
Software LBs announce a aggregated prefix

On failure, a more general prefix routed to Software LB
Performance

Traffic to SMux: 200k pkts/sec
Traffic to SMux: 400k pkts/sec
Traffic to HMux: 1.2m pkts/sec
Availability

![Graph showing availability and latency over time with markers for HMux1 failure and VIP1 traffic falling back to SMux at 38 msec.]

- VIP1 on HMux1
- VIP2 on HMux2
- VIP3 on SMux

Latency (msec) vs Time (msec)
Cost effective

![Graph showing the number of SMuxes for different VIP Traffic (Tbps) for Software LB and Duet.]

Duet reduces cost by 10-24x
Summary

• Advantages
  – Use commodity switches
  – Use small # software LB as backup
  – Low latency, low cost

• Disadvantages
  – Traffic redirection, link util high on path to LB switches
  – Fixed mapping of VIPs to DIPs
  – Does not exploit location of S, D