The Evolution of Traffic Routing in a Streaming World
A little about me

- Purdue University
- R&D Intern @ Bloomberg
- Program Manager Intern @ Microsoft
- Software Engineer @ Facebook
REAL-TIME INFRASTRUCTURE

Live-Video Messages & Reactions

Pam: Looking forward to learning about traffic routing!
Tim: SRECon is awesome!
Jon: First!

Anna is typing...

Typing Indicator
The Evolution

Collocated Architecture

Off-box Architecture

Full-Mesh Architecture
5000 ft view

Region A

Region B

Global Key-Value Store
Collocated Architecture
Collocated Architecture
Collocated Architecture

Problems

1. Shared Resources
2. Independent Deployments
3. Low Fault Tolerance
Collocated Architecture

[Diagram showing a collocated architecture with gateway and nexus components.]
Collocated Architecture

NEXUS

GATEWAY

GATEWAY

GATEWAY

GATEWAY

GATEWAY

NEXUS

GATEWAY

GATEWAY

NEXUS

GATEWAY

NEXUS
Off-box Architecture
Off-box Architecture

Diagram showing a series of GATEWAY boxes connected to NEXUS boxes, which are then connected to a database.
Off-box Architecture

Advantages

1. Customer Traffic Isolation
2. High Fault Tolerance
3. Dynamic Load Balancing
Advantages: Customer Traffic Isolation

- Isolation for customer A (E.g. live videos)
- Shared isolation for all other customers
Off-box Architecture

Advantages: High Fault Tolerance

Diagram showing the off-box architecture with GATEWAY nodes and NEXUS nodes connected.
Off-box Architecture

Advantages: High Fault Tolerance
Off-box Architecture

Advantages: High Fault Tolerance
Off-box Architecture

Advantages: Dynamic Load Balancing
Off-box Architecture

Advantages: Dynamic Load Balancing
Off-box Architecture

Advantages: Dynamic Load Balancing
Off-box Architecture

Advantages: Dynamic Load Balancing
Off-box Architecture

Advantages: Dynamic Load Balancing
Off-box Architecture

Challenges: Socket Connections

In a Full-Mesh connection:
Total Number of Connections: $G \times N$
Number of Connections per Nexus: $G$
Off-box Architecture

Challenges: Socket Connections

In a Full-Mesh connection:
Total Number of Connections: $G \times N$
Number of Connections per Nexus: $G$

If each Gateway Connected to only 'K' Nexuses:
Total Number of Connections: $G \times K$
Number of Connections per Nexus: $G \times K / N$

For our system, $K = 3$ provided desirable results.
Off-box Architecture

Challenges: Which Nexus to talk to?
Off-box Architecture

Challenges: Which Nexus to talk to?

- Nexus unavailable
- Fleet scaled up
- New Nexus Added
Off-box Architecture

Challenges: Which Nexus to talk to? A Naïve Solution
Off-box Architecture

Challenges: Which Nexus to talk to? A Naïve Solution

Unavailable (Due to deployment)
Off-box Architecture

Challenges: Which Nexus to talk to? A Naïve Solution

Unavailable
(Due to deployment)
Off-box Architecture

Challenges: Which Nexus to talk to? A Naïve Solution

Available Again
Off-box Architecture

Challenges: Which Nexus to talk to?

Conditions:

• Independent Gateway Decisions
• Even Distribution of Connections
• Minimal Disruptions
Rendezvous Hash

What is it?

Rendezvous hashing is an algorithm that allows clients to achieve distributed agreement on a set of 'k' options out of a possible set of 'n' options.
Rendezvous Hash

What is it?

Rendezvous hashing is an algorithm that allows clients to achieve distributed agreement on a set of 'k' options out of a possible set of 'n' options.
Rendezvous Hash

How does it work?

On Gateway $g_i$: 
For $n_j$ in $N$: 
\[ \text{weight of } n_j = \text{HASH}(g_i, n_j) \]
Connect to nexuses with the top 'k' weights values
Rendezvous Hash

How does it work?

On Gateway $g_i$:
For $n_j$ in $N$:
\[
\text{weight of } n_j = \text{HASH}(g_i, n_j)
\]
Connect to nexuses with the top $k$ weights values

\[
\begin{align*}
g_1 & \quad \text{GATEWAY} \quad n_1 = 78 \\
g_2 & \quad \text{GATEWAY} \quad n_2 = 13 \\
g_3 & \quad \text{GATEWAY} \quad n_3 = 44 \\
g_G & \quad \text{GATEWAY} \quad n_N = 15 \\
\end{align*}
\]

$k = 3$
Rendezvous Hash

How does it work?

On Gateway $g_i$:
For $n_j$ in $N$:

\[ \text{weight of } n_j = \text{HASH}(g_i, n_j) \]

Connect to nexuses with the top 'k' weights values
Rendezvous Hash

How does it work?

On Gateway \( g_i \):
For \( n_j \) in \( N \):
\[
\text{weight of } n_j = \text{HASH}(g_i, n_j)
\]
Connect to nexuses with the top 'k' weights values
Rendezvous Hash

A Note on the Hash and Distribution

On Gateway $g_i$:
For $n_j$ in $N$:

$$\text{weight}_\text{of}_n_j = \text{HASH}(g_i, n_j)$$

Connect to nexuses with the top `$k$' weights values
Rendezvous Hash

Minimal Disruption
Rendezvous Hash

Minimal Disruption

NEXUS GATEWAY GATEWAY GATEWAY NEXUS

\[ g_1 \]
\[ g_2 \]
\[ g_3 \]

<table>
<thead>
<tr>
<th>g_1</th>
<th>g_2</th>
<th>g_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>88</td>
<td>12</td>
<td>37</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>52</td>
</tr>
<tr>
<td>37</td>
<td>9</td>
<td>49</td>
</tr>
<tr>
<td>37</td>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>37</td>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>37</td>
<td>9</td>
<td>73</td>
</tr>
<tr>
<td>37</td>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>37</td>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>37</td>
<td>9</td>
<td>73</td>
</tr>
</tbody>
</table>

NEXUS

g_1 88 12 37 78 45 58 3

NEXUS

g_2 99 98 3 21 6 18 3

NEXUS

g_3 6 23 52 49 45 73 3

NEXUS
Rendezvous Hash

Minimal Disruption

The diagram illustrates a network of gateways and nodes labeled NEXUS. Each node is connected to one or more gateways, and the numbers associated with each connection represent values related to the rendezvous hash function. The gateways are connected in a hierarchical structure, with g₁, g₂, and g₃ as the top-level gateways.
Rendezvous Hash

Minimal Disruption
Rendezvous Hash

Minimal Disruption
Rendezvous Hash

Minimal Disruption

```
g_1 88
8

37

NEXUS

12
23

NEXUS

37
52

NEXUS

g_2 78
9

g_3

-45
-49

NEXUS

-21

NEXUS

58
6

g_3

18

NEXUS

45
73
```

NEXUS

GATEWAY

GATEWAY

GATEWAY
Off-box Architecture
Off-box Architecture

Traffic Routing

Diagram showing the routing architecture with NEXUS and GATEWAY components connected in a network.
Off-box Architecture

Traffic Routing

Diagram showing a gateway connected to three NEXUS nodes with bidirectional arrows.
Off-box Architecture

Traffic Routing

GATEWAY

NEXUS Streams: 100

NEXUS Streams: 5

NEXUS Streams: 100
Off-box Architecture

Traffic Routing

Streams: 100

Streams: 5

Streams: 100

After time 't'
Off-box Architecture

Traffic Routing: Naïve solution – pick lowest loaded

Before time 't':
- NEXUS (Streams: 100)
- NEXUS (Streams: 5)
- NEXUS (Streams: 100)

After time 't':
- NEXUS (Streams: 100)
- NEXUS (Streams: 205)
- NEXUS (Streams: 100)
Off-box Architecture

Traffic Routing: Naïve solution – Random host

After time 't'
Off-box Architecture

Traffic Routing: Lower of random two hosts

Streams: 100
Streams: 5
Streams: 100

Streams: 134
Streams: 137
Streams: 134

After time 't'
Off-box Architecture

Advantages

1. Customer Traffic Isolation
2. High Fault Tolerance
3. Dynamic Load Balancing
Full-Mesh Architecture
Full-Mesh Architecture
Full-Mesh Architecture

New Routing Capabilities

1. Sticky Routing
2. Virtual Isolation
Full-Mesh Architecture

Sticky Routing
Full-Mesh Architecture

Sticky Routing
Full-Mesh Architecture

Sticky Routing
Full-Mesh Architecture

Sticky Routing
Full-Mesh Architecture

Sticky Routing: Wins

15-20% reduction of requests to the Global Key-Value Store

5-10% Memory and CPU Savings on Bladerunner
Full-Mesh Architecture

Sticky Routing

How?
Full-Mesh Architecture

Sticky Routing: Rendezvous Hashing
Full-Mesh Architecture

Sticky Routing: Rendezvous Hashing

On any WarpGate:
Say we are routing feed $f_i$
Sticky Routing : Rendezvous Hashing

On any WarpGate:
Say we are routing feed $f_i$
For $br_j$ in Bladerunner:
  $weight_{of\;br_j} = HASH(f_i, br_j)$

Finally, route to Bladerunner with highest weight.
Sticky Routing: Rendezvous Hashing

On any WarpGate:
Say we are routing feed $f_i$
For $br_j$ in Bladerunner:
weight_of_br_j = HASH($f_i$, $br_j$)

Finally, route to Bladerunner with highest weight.
Full-Mesh Architecture

Sticky Routing

On any WarpGate:
Say we are routing feed $f_i$
For $br_j$ in Bladerunner:
\[ \text{weight}_{of\ br_j} = \text{HASH}( f_i, br_j ) \]
Finally, route to Bladerunner with highest weight.
Full-Mesh Architecture

Sticky Routing: Minimal Disruption

伟龙城域网

BLADERUNNER

WARPGATE

f₁  f₂

27  17

-73  -54

64  32

81  23

7  2
Full-Mesh Architecture

Sticky Routing: Minimal Disruption
Full-Mesh Architecture

Sticky Routing
Full-Mesh Architecture

New Routing Capabilities

1. Sticky Routing
2. Virtual Isolation
Looking back

Customer traffic Isolation in the off-box architecture

- Isolation for customer A (E.g. live videos)
- Shared isolation for all other customers
<table>
<thead>
<tr>
<th>Bladerunner 1</th>
<th>Lorem Ipsum</th>
<th>Dolor</th>
<th>Sit</th>
<th>Amet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bladerunner 2</td>
<td>Lorem Ipsum</td>
<td>Dolor</td>
<td>Sit</td>
<td>Amet</td>
</tr>
<tr>
<td>Bladerunner 3</td>
<td>Lorem Ipsum</td>
<td>Dolor</td>
<td>Sit</td>
<td>Amet</td>
</tr>
<tr>
<td>Bladerunner 4</td>
<td>Lorem Ipsum</td>
<td>Dolor</td>
<td>Sit</td>
<td>Amet</td>
</tr>
<tr>
<td>Bladerunner 5</td>
<td>Lorem Ipsum</td>
<td>Dolor</td>
<td>Sit</td>
<td>Amet</td>
</tr>
<tr>
<td>Bladerunner 6</td>
<td>Lorem Ipsum</td>
<td>Dolor</td>
<td>Sit</td>
<td>Amet</td>
</tr>
<tr>
<td>Bladerunner 7</td>
<td>Lorem Ipsum</td>
<td>Dolor</td>
<td>Sit</td>
<td>Amet</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Bladerunner 'b'</td>
<td>Lorem Ipsum</td>
<td>Dolor</td>
<td>Sit</td>
<td>Amet</td>
</tr>
</tbody>
</table>
Full-Mesh Architecture

Virtual Pools for Isolation

Virtual Pool of Lorem hosts
- Bladerunner 1
  - Lorem
- Bladerunner 6
  - Lorem

Virtual Pool of Ipsum hosts
- Bladerunner 2
  - Ipsum
- Bladerunner 4
  - Ipsum

Pool of all Bladerunner hosts
- Bladerunner 1: Lorem
- Bladerunner 2: Ipsum
- Bladerunner 3: Dolor
- Bladerunner 4: Ipsum
- Bladerunner 5: Sit
- Bladerunner 6: Lorem
- Bladerunner 7: Amet
- ...
Full-Mesh Architecture

Virtual Pools for Isolation

Noisy Neighbour

Resource Attribution

Maintenance Costs
Full-Mesh Architecture

Virtual Pools for Isolation

1. WarpGate Visibility
Full-Mesh Architecture

Virtual Pools for Isolation: WarpGate Visibility
Virtual Pools for Isolation: WarpGate Visibility

Full-Mesh Architecture

HOST LB FACTOR: 100000, HEAT STATUS: NORMAL,
CUSTOMER’S HANDLER: {
  IPSUM: {{ LB FACTOR: 35000, HEAT STATUS: LOW_UTIL },
  AMET: {{ LB FACTOR: 65000, HEAT STATUS: WARM },
}

Bladerunner 4  Ipsum  Amet
Full-Mesh Architecture

Virtual Pools for Isolation: WarpGate Visibility

Bladerunner 1
Lorem

Bladerunner 2
Ipsum

Bladerunner 3
Dolor

Bladerunner 4
Ipsum

Bladerunner 5
Amet

Bladerunner 6
Sit

Bladerunner 7
Lorem

Amet

...
Full-Mesh Architecture

Virtual Pools for Isolation

1. WarpGate Visibility

2. Virtual Pool placement
Full-Mesh Architecture

Virtual Pools for Isolation: Virtual Pool Placement

On each WarpGate:
Say we are to scale up $vpool_i$
Full-Mesh Architecture

Virtual Pools for Isolation: Virtual Pool Placement

On each WarpGate:
Say we are to scale up $vpool_i$
For $br_j$ in Bladerunner:

$weight\_of\_br_j = \text{HASH}(vpool_i, br_j)$
Full-Mesh Architecture

Virtual Pools for Isolation: Virtual Pool Placement

On each WarpGate:
Say we are to scale up $vpool_i$
For $br_j$ in Bladerunner:
\[
\text{weight}\_\text{of}\_br_j = \text{HASH}(vpool_i, br_j)
\]

Ignore weights of bladerunners that we don’t want to grow into.

Finally, upsize into the bladerunner host with the highest weight.
On each WarpGate:
Say we are to scale up $vpool_i$.
For $br_j$ in Bladerunner:
$weight_{of\ br_j} = HASH(vpool_i, br_j)$

Ignore weights of bladerunners that we don’t want to grow into.

Finally, upsize into the bladerunner host with the highest weight.
Full-Mesh Architecture

Virtual Pools for Isolation: Virtual Pool Placement

On each WarpGate:
Say we are to scale up $vpool_i$
For $br_j$ in Bladerunner:
$$\text{weight}_i = \text{HASH}(vpool_i, br_j)$$
Ignore weights of bladerunners that we don’t want to grow into.

Finally, upsize into the bladerunner host with the highest weight.
Full-Mesh Architecture

Virtual Pools for Isolation: Virtual Pool Placement

On each WarpGate:
Say we are to scale up vpool_i
For br_j in Bladerunner:
weight_of_br_j = HASH(vpool_i, br_j)

Ignore weights of bladerunners that we don't want to grow into.

Finally, upsize into the bladerunner host with the highest weight.
**Full-Mesh Architecture**

**Virtual Pools for Isolation: Virtual Pool Placement**

On each WarpGate:
Say we are to scale up vpool_i
For br_j in Bladerunner:
\[ \text{weight}_\text{of}_\text{br}_j = \text{HASH}(\text{vpool}_i, br_j) \]
Ignore weights of bladerunners that we don’t want to grow into.

Finally, upsize into the bladerunner host with the highest weight.

<table>
<thead>
<tr>
<th>Bladerunner</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>97</td>
</tr>
<tr>
<td>3</td>
<td>23</td>
</tr>
<tr>
<td>4</td>
<td>89</td>
</tr>
<tr>
<td>5</td>
<td>72</td>
</tr>
<tr>
<td>6</td>
<td>34</td>
</tr>
<tr>
<td>7</td>
<td>29</td>
</tr>
<tr>
<td>'b'</td>
<td>88</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Bladerunner</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lorem</td>
</tr>
<tr>
<td>2</td>
<td>Ipsum</td>
</tr>
<tr>
<td>3</td>
<td>Dolor</td>
</tr>
<tr>
<td>4</td>
<td>Ipsum</td>
</tr>
<tr>
<td>5</td>
<td>Ipsum</td>
</tr>
<tr>
<td>6</td>
<td>Lorem</td>
</tr>
<tr>
<td>7</td>
<td>Amet</td>
</tr>
<tr>
<td>'b'</td>
<td>Ipsum</td>
</tr>
</tbody>
</table>
Full-Mesh Architecture

Virtual Pools for Isolation: Virtual Pool Placement

On each WarpGate:
Say we are to scale down vpool_i
For br_j in Bladerunner:
weight_of_br_j = HASH(vpool_i, br_j)

Ignore weights of bladerunners that don’t have desired vpool_i

Finally, downsize the bladerunner host with the LOWEST weight in the virtual pool
Full-Mesh Architecture

Virtual Pools for Isolation: Virtual Pool Placement

On each WarpGate:
Say we are to scale down $vpool_i$
For $br_j$ in Bladerunner:

$$weight\_of\_br_j = HASH(vpool_i, br_j)$$

Ignore weights of bladerunners that don't have desired $vpool_i$

Finally, downsize the bladerunner host with the LOWEST weight in the virtual pool
Virtual Pools for Isolation: Virtual Pool Placement

On each WarpGate:
Say we are to scale down \( vpool_i \)
For \( br_j \) in Bladerunner:
\[
\text{weight of } br_j = \text{HASH}(vpool_i, br_j)
\]

**Ignore weights of bladerunners that don't have desired \( vpool_i \)**

Finally, downsize the bladerunner host with the **LOWEST weight** in the virtual pool
Virtual Pools for Isolation: Case Study of a Recent Event

Full-Mesh Architecture

Number of hosts 'Lorem' is on

Total inflight streams for 'Lorem'
Full-Mesh Architecture

Virtual Pools for Isolation

Noisy Neighbour

Resource Attribution

Maintenance Costs
The Evolution of Traffic Routing in a Streaming World

In Conclusion

1. Moving away from Collocation
The Evolution of Traffic Routing in a Streaming World

In Conclusion

1. Moving away from Collocation

2. Rendezvous Hashing
The Evolution of Traffic Routing in a Streaming World

In Conclusion

1. Moving away from Collocation

2. Rendezvous Hashing

3. Full Mesh Architecture
Thank you