Using PubSub For Scheduling in Azure SDN

Qi Zhang (Microsoft - Azure Networking)
Azure Networking

- SmartNIC/FPGA
- SONiC
- Virtual Networks
- Load Balancing
- VPN Services
- Firewall
- DDoS Protection
- DNS & Traffic Management
- DC Networks
- Regional Networks
- Optical Modules
- Software WAN
- Subsea Cables
- Terrestrial Fiber
- National Clouds
- Internet Peering
- ExpressRoute
- Acceleration for applications and content
- E2E monitoring (Network Watcher, Network Performance Monitoring)
Microsoft Global Network

One of the largest private networks in the world

- 8,000+ ISP sessions
- 130+ edge sites
- 44 ExpressRoute locations
- 33,000 miles of lit fiber
- SDN Managed (SWAN, OLS)

_DCs and Network sites not exhaustive_
Software Defined Networking (SDN)

Azure SDN
Basis of all NW virtualization in our datacenters

Control Plane
Centralized, hierarchical, highly scalable and available controllers

Data Plane
Host agent, drivers

Key to flexibility and scale is SDN
PubSub in SDN

• Scale:
  • 40+ regions, hundreds of DCs, millions of servers
  • millions of VNets and LBs

• Flexible, scalable and efficient scheduling between controllers and agents
• Publisher/Subscriber pattern
Virtual Network in Azure

Secure per customer virtual datacenter in the cloud

Instantiate and configure complex topologies in minutes

Rich security and networking services

Internet

VNet Peering

Cross premises Connectivity
CA-PA Mappings

Payload, including CA, is encapsulated and traverses the physical network.
PubSub for CA-PA Mapping

Challenges:
- Scale: hundreds K agents, millions of V Nets
- Scope: cluster, regional, global
- VNet size limit: 4K mappings -> 64K mappings, 500 peerings
- Provisioning Speed: minutes -> seconds
Scenario I: Global Peering

Region A / VNET A

VNet Controller

Agent

PubSub

Publish mappings for VNET A

Subscribe mappings for VNET A

Region B / VNET B

VNet Controller

Agent

PubSub

Subscribe mappings for VNET A from remote PubSub

Subscribe mappings for VNET B from remote PubSub

Publish mappings for VNET B
Scenario II: DataExfil

Service Tunnel Policy

```json
{
    id: "policy-123",
    service: "xstore",
    subscription: ":(guid),",
    accounts: ["users",
               "wiki,"],
    storage_type: "blob",
    access: "rw"
}
```

**Resource “Metadata”**

```
METADATA (resource A):
{
    subscription: "(guid),",
    account: "users",
    storage_type: "blob"
}
```

**Resource “Metadata”**

```
METADATA (resource B):
{
    subscription: "(guid),",
    account: "users",
    storage_type: "table"
}
```

**Resource “Metadata”**

```
METADATA (resource C):
{
    subscription: "(guid),",
    account: "wikimain",
    storage_type: "blob"
}
```

![Diagram showing resource access control with 'ALLOW' and 'BLOCK' markers]
Overview

• Persisted KV Store
  • Hierarchical name space
  • Set watcher on a node
    • Single watcher
    • Bulk watcher

• Interfaces
  • Publish (batch/multi supported)
  • Subscribe
  • Notification
  • Query

• State Update/Delivery
  • Initial state
  • Subsequent state updates
4 Microservices:
- Stateless Service
  - Routing Service
  - Notification Service
- Stateful Service
  - Selector Service
  - Madari Service

Scale to expected Data requirements. Add more instances as data increases.
Partition Key

4 Microservices:

Stateless Service
- Routing Service
- Notification Service

Stateful Service
- Selector Service
- Madari Service

Publisher (Vnet Controller)
1. PK: /Vnet/{VnetId1}, Path: /mappings/ipv4/{CA1}
   Data (bond message): {PA1}
2. PK: /Vnet/{VnetId1}, Path: /mappings/ipv4/{CA1}
   Data (bond message): {PA1}
3. PK: /Vnet/{VnetId1}, Path: /mappings/ipv4/{CA1}
   Data (bond message): {PA1}
4. PK: /Vnet/{VnetId2}, Path: /mappings/ipv4/{CA1}
   Data (bond message): {PA1}
5. PK: /Vnet/{VnetId2}, Path: /mappings/ipv4/{CA1}
   Data (bond message): {PA1}
6. PK: /Vnet/{VnetId2}, Path: /mappings/ipv4/{CA1}
   Data (bond message): {PA1}

Subscriber Agent
1. PK: /Vnet/{VnetId2}, Path: /mappings/ipv4/{CA1}
2. PK: /Vnet/{VnetId2}, Path: /mappings/ipv4/{CA1}
3. PK: /Vnet/{VnetId2}, Path: /mappings/ipv4/{CA1}
4. PK: /Vnet/{VnetId2}, Path: /mappings/ipv4/{CA1}
5. PK: /Vnet/{VnetId2}, Path: /mappings/ipv4/{CA1}
6. PK: /Vnet/{VnetId2}, Path: /mappings/ipv4/{CA1}

Scale to expected Data requirements. Add more instances as data increases

SDN PubSub Service
AddPartitionKey("baz")

<table>
<thead>
<tr>
<th>Partition Key</th>
<th>Madari Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;foo&quot;</td>
<td>MadariService_01</td>
</tr>
<tr>
<td>&quot;bar&quot;</td>
<td>MadariService_02</td>
</tr>
<tr>
<td>.....</td>
<td>.....</td>
</tr>
<tr>
<td>&quot;baz&quot;</td>
<td>MadariService_01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Madari Instance</th>
<th>Total Data Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>MadariService_01</td>
<td>1.05G</td>
</tr>
<tr>
<td>MadariService_02</td>
<td>1.9G</td>
</tr>
<tr>
<td>MadariService_03</td>
<td>1.6G</td>
</tr>
</tbody>
</table>

Selector Service

Madari Selector Service: Data Partitioning
Subscription through Notification Service
Service Fabric Ring

- Service Fabric ring
  - Multiple PaaS tenants form a Service Fabric ring
  - Service Fabric ring is on a VNET

- PubSub as Service Fabric application
  - Routing Service/Notification Service
    - Stateless
    - On every node
  - MadariService/MadariSelectorService(s)
    - Stateful
    - Min 3, target 7
Client Libraries

- Managed Libraries
  - Madari.ClientLibrary
  - Publishing through WCF channel
  - Reliable Publisher
    - IMOS-based publishers
    - User implements:
      - Commit hooks
      - Handler
    - Nuget package:
      Madari.ReliablePublisher.RSL
      Madari.ReliablePublisher.ServiceFabric

- Native Libraries
  - Publish
    - Nuget package:
      Madari.MadariFrontEnd.Native
  - Subscribe
    - Nuget package:
      Madari.Subscriber.Native
Hierarchical PubSub Infrastructure

Resource Scope => PubSub Service Scope

<table>
<thead>
<tr>
<th>Resource</th>
<th>Scope</th>
<th>Publisher</th>
<th>Subscriber</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA-PA mapping</td>
<td>regional</td>
<td>VNet Controller</td>
<td>Agent</td>
</tr>
<tr>
<td>DataExfil policy</td>
<td>global</td>
<td>NRP</td>
<td>Agent</td>
</tr>
</tbody>
</table>

![Diagram showing hierarchical PubSub infrastructure with nodes for Global PubSub, Regional PubSub, and DataExfil policy.]
Global PubSub

Region A
- PubSub (AZ01)
- PubSub (AZ02)
- PubSub (AZ03)

Region B
- PubSub (AZ01)
- PubSub (AZ02)
- PubSub (AZ03)
Publish Policy – No Replication (Sync)
Replication Service

Operation Tracking Table

<table>
<thead>
<tr>
<th>Op Id</th>
<th>Status</th>
<th>Operation</th>
<th>Replication Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>Replicated</td>
<td>[add] /DataExfil/Policies/Policy1 {Dest1:Y, Dest2:Y, Dest3:Y }</td>
<td></td>
</tr>
<tr>
<td>1002</td>
<td>Replicating</td>
<td>[update] /DataExfil/Policies/Policy1  {Dest1:Y, Dest2:N, Dest3:Y }</td>
<td></td>
</tr>
<tr>
<td>1003</td>
<td>Committed</td>
<td>[remove] /DataExfil/Policies/Policy1   {Dest1:N, Dest2:N, Dest3:N }</td>
<td></td>
</tr>
</tbody>
</table>
Global SF Ring
Major Performance KPIs

• 15 partitions

<table>
<thead>
<tr>
<th>KPI</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write throughput</td>
<td>10k req/s</td>
</tr>
<tr>
<td>Read throughput</td>
<td>42k req/s</td>
</tr>
<tr>
<td>End to End latency</td>
<td>10ms/300ms (50%/99%)</td>
</tr>
<tr>
<td>Max subscribers</td>
<td>500K</td>
</tr>
</tbody>
</table>

• In a large region:
  • < 300k agents
  • < 100K VNets
  • ~1k read/sec, ~200 write/sec
Work in Progress

- Accelerating read flow

- End to end validation
Q & A

Thank you!