Arion control plane

Futurewei Cloud Lab

2/2022

Background

- Technical trend:
  - Elastic in both networking functionalities and cost
  - Thinner host/agent or even baremetal, offshore networking functionalities
  - DP-triggered or DP-like CP, for example session missed on-demand lookups

- Realistic challenges
  - Host/ACA's growing rules and resources
    - Relationship between pushing static rules and dynamically lookup from upstream
  - NCM stress as a central lookup service
    - On-demand roundtrip latency and concurrency
  - Some features may fit better on GW rather than spreading on each host
    - SG
  - GW brings more coarse-grained control in aggregated traffic
    - Rate limiting (ingress, egress) per VPC
    - QoS adjustment across multiple applications per VPC, or per region (cross VPCs)

Scenarios and workload

Phase I - target and constraints

For each Arion cluster

Target support:

Compute Node: 100K VMs: 2M

VNI: 20 Flow rules: Neighbor rules

Mapping rules: dstip+vni -> CN IP, total rule capacity: 2M(?)

Arion Master capacity:

256G physical memory, 1T disk

ArionWing capacity:

100G Nic card, 32G physical memory, 512G disk,

12(24) ArionWings

Multiple eBPF tables in each ArionWing, each ArionWing covers rules for 25K cn/500k vm(or 12.5K cn/250K vm); sustains multi ArionWing failure and leaves space for complex rule handling(e.g. ACL, SG, cross VPC, etc).

Target supported throughput capability:

1Tbps(2TGbps) throughput and sustains temporal multiple ArionWing failures.

Notes: If we use 400G Nic card, the capacity increases to 4Tbps/8Tbps in above calculations.

Notes:

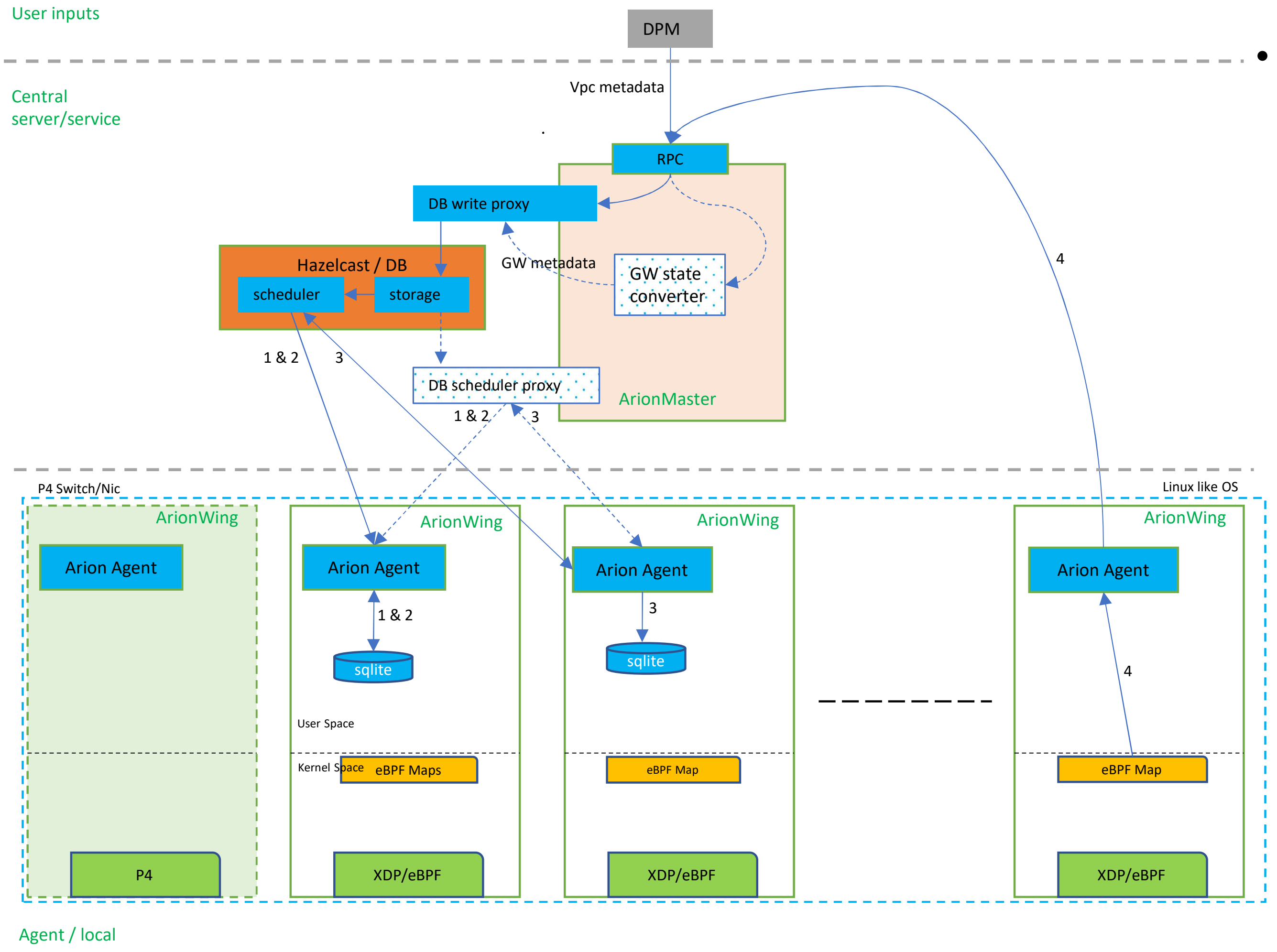
T = total Arion Wings(min 6)
N = Arion Wings in each group(default 3)
G = T/N, total groups(min 2)

1. **Every N Arion Wings form a group, which has the same flow control rules;**
2. **Total flow control rules are sharding into G subsets.**
3. This is 1/10 of the designed deployment, which could have 120(240) Arion Wings with 10T(20T) bps throughput, or 40T/80Tbps with 400G Nic for 1M compute nodes in VPC.

Roadmap

- Phase 1 - End of June
  - Achieve hover-board GW scenario (specific feature of vpc neighbor connectivity, and limited traffic)
  - GW eBpf top-down programming
- Post phase 1 (phase 1.5) - End of September
  - Performance (clients, qps and latency)
  - 100M+ metadata, for vpc policies (phase 1 is neighbors) that offloaded to Arion
  - Sub-ms session lookup
    - Local lookup definately less than 1ms
    - Remote lookup, around 1ms, best effort to achieve less than 1ms
- Phase 2 - End of December
  - Reuse/extend the framework to manage much higher-throughput and much higher data volume GW units
  - Enable a stateful scenario of GW

Arion Architecture



- Design philosophy
  - Phases
    - Phase 1 module — Phase 1 workflow
    - Phase 2 module — Phase 2 workflow
  - Master + db
    - Provides reliable data persistence and high-performance data lookup for GW data
    - high concurrency in different types of jobs
  - Communication (master + db <-> wings)
    - push GW (eBPF) goal states (#1)
    - reconcile/restore (#2)
    - on-demand lookups (#3)
    - close-loop status reporting (#4)
  - Near-target lookup, but with fallback
    - Push down the function of cache and lookup to Arion Wing, if need to guarantee <1ms response
    - Best effort to minimize roundtrip latency between ArionWing and ArionMaster/DB channel when query remotely
    - ArionWing (with eBPF) will decide when query locally and when query remotely, the criteria and percentage between them

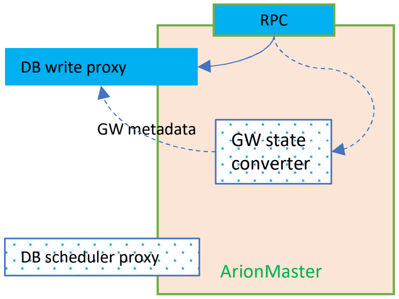
Modules

Hazelcast (as vpc DB, and can be replaced with other DBs)

- Storage – vpc metadata persistence and lookup, for Arion phase 1 the table that GW is interested is vpc neighbor table (which already provided, doesn't need new table or new metadata format)
- Job scheduler of db query, watch, get etc.

Arion master

- Rpc
  - The only place to write to DB is through Arion master rpc call
  - Includes
    - For DPM to write vpc metadata
    - For Arion Wings to report programming status
- Communication channels (with Arion Wings)
  - DB notify
    - push GW (eBPF) goal states (#1)
    - reconcile/restore (#2)
  - DB get: on-demand lookups (#3)
  - Call rpc of Arion Master: close-loop status reporting (#4)
- High concurrency job scheduling
- Reserve 2 wrapper layers to hook with general DBs
  - DB write proxy, implements writing functions
  - DB scheduler proxy
    - hooked with DB internal callbacks
    - to implement an abstract layer of notify functionalities like watch and get



DB

- Goals
    - Stores entire region vpc metadata
        - Vpc neighbors is shared (stored only once) among Alcor/DPM and Arion, so Arion phase 1 (neighbor rules) doesn’t need GW new table/db-schema in Hazelcast
    - Performance (clients, qps and latency)
        - For regular notifying and syncing vpc updates (version by version) to Arion-wings
            - < 1ms
        - For Arion-wings reconcile
            - New slice deployment
                - ~1G data, < 10 s
                - ~3G data, < 30 s
            - Arion-wing crash and restore, will trigger a series of version updates
        - For Arion-wings on-demand lookups
            - < 120 clients, 260k qps, with <1ms latency

- General DB to integrate with
    - Need to provide hooks of
        - watch (new records, or record updates notification/callback)
        - get (query, wrap around and reply in high performance manner)

Arion Master

- Deploy 1 (service) per partition
  - each client connects to master is a gateway node
  - Depending on the throughput capacity per gateway node, the client number are 6 - 240

- Goals
  - 100M+ metadata, for vpc policies (phase 1 is neighbors) that offloaded to Arion
  - Sub-ms session lookup from db
    - local end time – local start time
    - remote receive on-demand reply – remote send request
  - 1M+ qps
    - Measured local
    - Measured remotely

- Non-goals (future planning)
  - Better balancing of different levels of cache

Arion Wing

- Goals
  - Turn upstream db schema to DP operations
  - Local db persistence (sqlite)
    - to provide most efficient on-demand lookup
    - also provides last-known-good version to restore from
  - Reconcile from local data persistence and remote db, the total data amount per Arion Wing is a few GBs
  - Sub-ms session on-demand lookup
    - Start time is when packet (initial connection) session missed, and put on-hold by data plane, need to see what is the behavior from eBpf
    - End time is when lookup done, and release packet from dp

- Non-goals (future planning)
  - Balancing and switching different session pools, like hot warm and cold