High Availability Load Balancing in the Agile Infrastructure

HEPiX Bologna, April 2013

Vaggelis Atlidakis, CERN IT-PES/PS
Ignacio Reguero, CERN IT-PES/PS
Outline

- Core Concepts
- Service Manager's Concerns
- DNS Load Balancing at CERN
- HA Services in the AI
- OpenStack LBaaS within Quantum
- CERN network restrictions
- Conclusions
Core Concepts

Load Balancing:
- Scale a single service by spreading it to multiple back-end nodes.

High Availability:
- From an end user's perspective, service should be always functional.
- Service should be available when some back-end or front-end nodes are unhealthy.
Service Manager Concerns

Implement high availability at the application Layer:

- Service components should interact without any single point of failure.
- Replicate physical nodes among independent subnets.
- Replicate VMs in different availability zones.

Service components are expected to fail:

- Hardware failures (HDD, Switches, NIC, Electricity, etc.).
- Software failures (Bugs).
- Human errors.
We use a client/server architecture:
- LBD Master: Server which reports to DNS service.
- LB Client: Daemon running in the hosts.

- LB Clients in the hosts provide LBD Master with load metrics as well as availability checks (SNMP communication).
- LBD Master decides which IP should be pointed by an LB alias.
- The LBD Master sends Dynamic DNS requests to update the IP address pointed by the LB alias.
- The LBD Master uses a fail-over slave server for high availability (The slave is consistent with the master).
High Availability Load Balancing in the Agile Infrastructure

**DNS LB**

- **User** queries for `aiplus6` which resolves to `137.138.241.29`.
- The DNS server updates its records to point to this IP address.
- `aiplus6` is a LB alias.
- Members receive load metrics from the LBD Master.
- The LBD Slave gets heartbeat file.

**LBD Master** receives SNMP Load Metric inputs: -1, 30, 10.
• Able to provide a general service for almost 150 different aliases.

• It does so without network traffic bottleneck.
- No session persistence; needed for web applications that are statefull.

- No virtual IPs supported.

- Manual action required to define new LB aliases in the DNS service: Network engineer required to define new aliases.

- Delegation only effective once the LB alias has been created.
Components

- VMs in our OpenStack private cloud (KVM, HyperV).
- Configuration Management with puppet.
- Node Classification with foreman.
- Monitoring with Lemon.
- LBD
- HAProxy (Application Layer Load Balancer)
HAProxy Background

- HAProxy is a free, very fast and reliable solution offering Load Balancing.

- It is a layer-7 Load Balancer capable of support proxying for TCP and HTTP-based applications.

- It can operate under a pass-through or redirect reverse proxying configuration.
HAProxy Background

- HAPerxy is flexible to configure and supports various Load Balancing policies:
  - round-robin
  - weighted round-robin
  - leastconn
  - source-IP (affinity)
  - rdp-cookie (persistence)

- Availability check.

- It is the software Load Balancer used in Grizzly release of OpenStack.
Recommended Scenario

- Service manager deploys and replicates back-end nodes in different availability zones.
- Chooses HAProxy as the actual load balancer for his application.
- Deploys instances running front-end HAProxy balancers in different availability zones.
- Creates an alias for his service.
- Front-ends report to LB Master.
- DNS resolves service IP to a healthy front-end, which redirect traffic accordingly.
HA Services in the Agile Infrastructure

User ➔ Query for aisusie
188.184.132.115 ➔ DNS

DDNS update aisusie to
188.184.132.115

Zone 1
- aisusie LB members
- HAProxy LB1

Zone 2
- HAProxy LB2

Zone 3
- SNMP Load Metrics
- LBD Master ➔ LBD Slave

Back-end nodes
- Worker 1
- Worker 2
Intoduction to LBaaS

- OpenStack user should be able to create Load Balancers from the Dashboard (Horizon).

- Infrastructure should provide API for this functionality.

- User should be able to configure the Load balancing service from the API.

- No access to the actual load balancer.
OpenStack LBaaS (Equilibrium)

- Implemented in Python, from Mirantis, using OpenStack templates.
  - Based on OpenStack common code.
  - Uses OpenStack Services.

- Integrated with OpenStack Horizon GUI.

- Referenced by Quantum OpenStack network component.
OpenStack LBaaS (Equilibrium)
Key Features

- REST API for cloud admins: manage a pool of HW and SW load-balancing appliances.

- REST API for OpenStack tenants: load balancing as a service, multi-tenancy support and isolation.

- Drivers supporting load balancers from different vendors such as HAProxy (sw) and Cisco ACE (hw).
Equilibrium Tenant API (1)

- **Load Balancers API:**
  - Get a List of Existing LB.
  - Create Load Balancer Instance.
  - Delete Load Balancer Instance.
  - Update Load Balancer Instance.
  - Get Load Balancer Instance Detailed Information.
  - Get Load Balancer status.
  - Get Load Balancing statistics.

- **Load Balancer Node API:**
  - Add Node to Existing LB.
  - Get List of Nodes.
  - Delete Node from Load Balancer.
  - Update Node in Load Balancer.
  - Change state of Node.
Equilibrium Tenant API (2)

• Health Monitoring API:
  ➔ Get List of Probes Attached to Load Balancer.
  ➔ Add Probe to Load Balancer.
  ➔ Delete Probe from Load Balancer.

• Session Persistence API:
  ➔ Get a List of Session Persistence Configured for Load Balancer.
  ➔ Add session persistence rule for Load Balancer.
  ➔ Delete Configured Stickiness.

• Configuration:
  ➔ Get a List of Supported Load Balancing Protocols.
  ➔ Get a List of Supported Load Balancing Algorithms.
- In our CERN private Cloud we intend to provide LBaaS.
- We start evaluating Mirantis Equilibrium.
- Equilibrium meets our needs.
Restrictions due to CERN's network structure:

- Virtual IPs in CERN's network cannot move out of a network service (normally corresponding to a subnet).
- They should all appear behind the same network box (switch or other).
- Automatic registration of IP aliases not supported (requires human intervention).
Conclusion – Future Plans

For LBaaS we need:

- Back-end nodes running equilibrium instances.
- A pool of sw (or hw) Load Balancers, running HAProxy instances.
- A service manager maintaining this pool and the puppet modules for equilibrium.

LBaaS will support:

- Session persistence.
- Virtual IPs.
- Unified Configuration of LBs with Rest-full API.
- Application Layer LB.
- Availability check.
Thank you for your attention!

Questions???