Microservice Powered Orchestration

Huabing Zhao  ZTE, System Engineer, Network Management & Service, OPEN-O Common Service PTL
zhao.huabing@zte.com.cn
Zhaoxing Meng ZTE, NFV&SDN Architect, Network Management & Service, OPEN-O Common Service PTL
zhaoxing.meng1@zte.com.cn
Agenda

• Why Microservice at OPEN-O
• Challenges of Microservice
• MSB(Microservice Bus) Solution
• What can MSB bring to ONAP
The microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API.

-Martin Fowler
Why choose Microservice Architecture?

How to integrate existing seed codes in different technical stack?
- We didn’t start from scratch
- A dozen of existing seed codes repos
- Ambitious release plan

How to make orchestration reliable and scalable?
- OPEN-O is a large, complex software system
- Each component may have different resource requirement
- Each component may have different working load

How to build an OPEN community?
- We have various members and we are expecting more joining in
- Each organization has its own tech Stack
We had made an ambitions plan for SUN Release

**June 8, 2016**
Kick-Off

**August 11, 2016**
Planning

**Sept 1, 2016**
API Freeze

**Sept 29, 2016**
RC0

**Sept 15, 2016**
Code Freeze

**Nov 3, 2016**
Sun Release
Challenge of Integration

We get bigger challenge for ONAP integration
Build an open community so that everyone can enjoy the party.
Challenges of Microservice Architecture

Microservice Architecture comes at a price: Complexity

How do the clients application access the back end services?

How do the client or another service - discover the location of a service instance?
Direct Client-to-Microservice Communication?

This approach has some problems:

- Add complexity to client codes
- Nightmare for firewall configuration
- Coupling of client and individual services
- Cross-domain issue for web app
Solution: Service Gateway

Service gateway hides the complexity

- Simplify the client codes.
- Reduce request roundtrips
- Provide API management
- Solve cross-domain issue for web app
How to find the service?

In order to access a service, you need to know the exact endpoint (IP & Port)

“Traditional” application

- Service endpoint doesn’t change a lot
- Consumer can get the endpoint from configuration files

Microservice application

- The IP & port is dynamically allocated
- IP & port changes along with the scaling/ updating/ self-healing of service instances
Solution: Service Registration & Discovery

Service Registration:
➢ Service providers register themselves to the registry when start up
➢ Update service information when service instances change

Service Discovery:
➢ Service consumers query registry to find the locations of service
➢ Two approaches: Server-side discovery & Client-side discovery
OPEN-O Microservice Solution: High Level Architecture

**Service Discovery (DNS Server)**
- **Listen to service change**
- **Register Heartbeat**
- **Unregister**
- **DNS Search**
- **Access Service**
- **Service Provider Instance A**
- **Service Provider Instance B**

**Service Consumer**
- **Access Service (Client-side discovery)**
- **Access Service (Server-side discovery)**

**Registration Proxy**
- **Listen**
- **Register**

**L7 Service Updater**
- **Listen to service change**
- **Update Service Registry**

**L4 Service Updater**
- **Modify and Reload**

**Cache**
- **Query Service Registry**

**Service Gateway**
- **Listen to service change**
- **Load Balance**

**Access Service**
- **Modify and Reload**

**Request Routing**
- **Service Discovery**

**DNS Search**
OPEN-O Microservice Solution: MSB Components

Microservices

- Docker Cluster
  - Service A
  - Service B
- Other Cluster (VM, Mesos, K8S, Swarm ...)
  - Service A
  - Service B

Docker events

Service lifecycle events

Healthy Check

Register

Forward registration request

Registration Proxy

- Docker Listener
  - DockerProxy
  - Discovery Client
- Other Listeners
  - OtherProxy
  - Discovery Client

Service Discovery

- Service Discovery Client
  - Discovery Management
  - Discovery Client
  - Discovery Server

Forward registration request

Service Gateway

- Service Gateway
  - OpenResty
  - L7 Service Updater
  - L4 Service Updater
  - Cache

Query

Update

External Systems

- 3-party App
- UI Portal
MSB Features-High Availability

**Access Layer**
- Load balancer (DNS Server/LVS etc.) in the front end
- Service gateway cluster to avoid SPOF of service gateway

**Service Layer**
- Service gateway as the load balancer for services
- Deploy multiple service instances to avoid SPOF of service
**MSB Features - Separated gateway for External and Internal Routing**

**External service gateway**
- Expose the services (Rest API, UI pages, etc.) which need to be accessed by external systems
- Solve the cross-domain issue for web app
- Stricter access control
- Adaption between external API and internal service

**Internal API gateway (router)**
- Routing and load balancing of the API calls within the system
- Less control in trusted zone
- Light weight communication protocol

Additional features:
- Stricter access control
- Protocol translation (e.g., https -> http)
- Expose the services (Rest API, UI pages, etc.) which need to be accessed by external systems
- Solve the cross-domain issue for web app
- Stricter access control
- Adaption between external API and internal service

Can add more gateways according to deployment scenarios.
• Extendable architecture for adding functionality
  ➢ Auth: add auth to APIs, integrated with Openstack keystone
  ➢ Driver routing: add driver specify routing logic for devices
  ➢ Logging: API calling logging
  ➢ Service health monitoring
  ➢ ACL, API Analytics, Transformations
  ➢ Anything: new functionality can be added on demand by plugins
MSB Features - Service Healthy Monitoring

- JVM Memory Distribution
  - Total
  - Non-Heap
  - Heap

- JVM Memory Usage
  - Code-Cache
  - Eden-Space
  - Perm-Gen
  - Survivor-Space
  - Tenured-Gen

- Thread Distribution
  - Blocked
  - Waiting
  - Timed-waiting
  - Runnable

- HTTP Access
  - Request interface traffic rankings

- HTTP traffic rankings
MSB Features-API Monitoring

- **Realtime Connection Status**
  - Active
  - Waiting
  - Writing
  - Reading

- **Pending Request Status**
  - Sent
  - Received
  - Tolerated

- **Historical Request Count**


![Graphs showing real-time and historical API monitoring metrics](image-url)
How MSB may fit into ONAP (Service Discovery & Routing)

Before:

```
"aaiEndpoint": "https://c1.vm1.aai simplesdemo. openecomp. org:8443",
"adaptersCompletionsprocessEndpoint": "http://mao:8080/CompleteMaoProcess",
"adaptersDBEndpoint": "http://mao:8080/db/adapters/RequestDBAdapter",
"adaptersSnoEndpoint": "http://mao:8080/adapters/SDNCApi",
"adaptersTenantEndpoint": "http://mao:8080/tenants/TenantAdapter",
"workflowSnoEventCallback": "http://mao:8080/mao/SDNCApiEventCallbackService",
"adaptersNetworkEndpoint": "http://mao:8080/networks/NetworkAdapter",
"adaptersNetworkRestEndpoint": "http://mao:8080/networks/rest/v1/networks",
"adaptersVnfAsyncEndpoint": "http://mao:8080/mao/vnfAdapterAsync",
"workflowVnfAsyncDeleteCallback": "http://mao:8080/mao/vnfAdapterNotify",
"workflowVnfAsyncCreateCallback": "http://mao:8080/mao/vnfAdapterNotify",
"adaptersVnfRestEndpoint": "http://mao:8080/vnfa/rest/v1/vnfa",
```

After:

```
"apigateway": "https://apigateway.onap.org:80"
```

How to call service:

GET https://apigateway.onap.org/api/aai/v8/cloud-infrastructure/cloud-regions/cloud-region/{cloud-owner}/{cloud-region-id}

API gateway routes the request to:

GET https://c1.vm1.aai simplesdemo. openecomp. org:8443/aai/v8/cloud-infrastructure/cloud-regions/cloud-region/{cloud-owner}/{cloud-region-id}

Using a configuration file, we might have problems on scaling, failover and update.

MSB as the single entry point

MSB handles the service discovery & routing & LB
How MSB may fit into ONAP (reverse proxy for web app)

**Before:**
- The business logic (rest service) forwarder must be added to the front end server.
- Solve the cross-domain issue causing coupling of business logic and UI pages.

**After:**
- Service gateway to solve cross-domain issue.
- Cache for static resources (page, picture).
- Clearer boundary between UI and business logic.
Thank You

www.onap.org