MSB to Support for Carrier Grade ONAP Microservice Architecture

Huabing Zhao, PTL of MSB Project, ZTE
ONAP Architecture Principle: Microservices

ONAP Architecture Principle: ONAP modules should be designed as **microservices**: service-based with clear, concise function addressed by each service with loose coupling.

In Amsterdam Release, MSB provides infrastructure to support ONAP Microservices architecture.

- Registration/Discovery
- API Gateway
- JAVA SDK
MSB Amsterdam Overview

- **Registry**: Service information storage, MSB uses Consul as the service registry.
- **MSB Discovery**: Provides REST APIs for service discovery and registration
- **API Gateway**: Provides service request routing and load balancing. It can be deployed as an External Gateway which is responsible for exposing ONAP microservices to external system or an Internal Gateway which can be used for the inter-service communication inside ONAP
- **MSB SDK**: Java SDK for point to point communication
MSB Amsterdam Design Principle

Providing Microservice Infrastructure for ONAP in a transparent way, with no or minimum modification on the application side. This principle is reflected in the design and implementation of the MSB components.

Example 1: Service registration by Kube2MSB Registrator:

- OOM(Kubernetes) deploy/start/stop ONAP components.
- Registrator watches the kubernetes pod event.
- Registrator registers service endpoint info to MSB. It also updates the service info to MSB when ONAP components are stopped/restarted/scaled by OOM.

No registration codes needed on the application side.
Providing Microservice Infrastructure for ONAP in a transparent way, with no or minimum modification on the application side. This principle is reflected in the design and implementation of the MSB components.

Example 2: Inter-service communication via Internal Gateway

- Internal Gateway handles the communication details including service discovery, load balancing, retries, etc.
- Multiple Internal Gateways can be deployed as a cluster to share the workload, which is a similar concept to service mesh, but not one proxy for each service instance.
OMSA is the vision of ONAP Microservice Architecture to support Carrier-Grade requirements of ONAP Microservices, which includes Service Orchestration, Service Discovery, Inter-service Communication, Service Governance and Service Monitoring and External API Gateway.

**Need enhancement for carrier grade ONAP:**

- Reliability and security of RPC between microservices
- Visibility of microservices (metrics, distributed tracing)
- Manageability of microservices (policy, API management, Configure management)
- Performance (Cache, Latency aware load balancing)

Note: this diagram is a functional view of OMSA, which is not mapped to specific projects.
Enhance Microservice Infrastructure to Support Carrier Grade ONAP

MSB plans to provide service mesh to enhance

- Reliability and security of RPC between microservices
- Visibility of microservices (metrics, distributed tracing)
- Manageability of microservices (policy, API management, Configure management)
- Performance (Cache, Latency aware load balancing)
Brief Introduction of Service Mesh

Amsterdam
Amsterdam

App container
Amsterdam

App container

TLS
Amsterdam
Amsterdam
Amsterdam

App container

- TLS
- retries
- metrics
Beijing
Separation of concerns

- The remote communication logic is moved into the proxy and deployed as “sidecar”
- Developers only need to take care of the business logic

The Linux Foundation
A service mesh is a dedicated infrastructure layer for handling service-to-service communication. It’s responsible for the reliable delivery of requests through the complex topology of services that comprise a modern, cloud native application. In practice, the service mesh is typically implemented as an array of lightweight network proxies that are deployed alongside application code, without the application needing to be aware.

-William Morgan
MSB Plan for R2 and Beyond

Provides Service Mesh to support Carrie Grade, Microservice-Based ONAP

- **Stability and Reliability**: Reliable communication with retries and circuit breaker
- **Security**: Secured communication with TLS
- **Performance**: Latency aware load balancing with warm cache
- **Observability**: Metrics measurement and distributed tracing without instrumenting application
- **Manageability**: Routing rule and rate limiting enforcement
- **Testability**: Fault injection to test resilience of ONAP
MSB Architecture for R2 and Forward

- MSB Discovery is plugged into Pilot as a service discovery source
- MSB Internal API Gateway is enhanced as a sidecar proxy to address the Carrier Grade challenges
- It can be deployed as the Internal API Gateway that is a cluster shared by multiple service instances or the sidecar that is one per service instance
- MSB API Gateway is used to expose ONAP services to external system
- Add MSB Portal to control plane to provide service Catalog, swagger UI of Restful API, service mesh configuration (long term goal), etc.