Satifying-Cactus Roadmap

An overview of the Roadmap was also presented at CORD Build (November 7-9, 2017). See the attached presentation.

Priorities

CORD 4.0 stabilized service developer interfaces. The next goal is to build out CORD's service portfolio:

- Upgrade all current R/E/M-CORD services to the 4.0 API
- Integrate latest access peripherals – VOLTHA and xRAN
- Expand service portfolio to include micro-services – Kubernetes
- On-Board other VNFs into CORD – ONAP
- Streamline service on-boarding based on experience

CORD 4.0 refactored build system to improve developer workflow. The next goal is to exploit this flexibility to improve operator workflow:

- Automate the build-and-install process for physical PODs, including discovery and configuration of the POD switching fabric
- Make it easy to specify (and change) service profiles independent from configuring the underlying platform
- Improve lifecycle management capabilities to include in-service-software-upgrade of the CORD control plane
- Demonstrate how CORD can leverage available infrastructure rather than require that a POD be build on top of bare metal

All this work is driven by the following deliverables:

- Multi-Access Edge Cloud
    - R/E/M-CORD Services running on the same platform
    - Includes VOLTHA and xRAN access peripherals

- Managed XGS-PON Peripheral
    - Includes VOLTHA
    - Light-and-Right R-CORD (Kubernetes-based)
    - Includes OSAM (Open Source Access Manager)

- Integrate CORD and ONAP
    - Use Case 1: OSAM
    - Use Case 2: E-CORD / MSO
    - Use Case 3: A-CORD / DCAE

Platform Roadmap

Build System

- Improved Development Workflow
    - Easier debugging -- ElasticStack & structured logging
    - Easier to develop use cases à Decouple profiles from platform
- Improved Operator Workflow
    - Fast, foolproof install process
        - Install all containers from Docker Hub onto K8S
        - Install a generic CORD system, use dynamic service onboarding
    - Modular, flexible CORD
        - Use existing infrastructure (e.g., OpenStack, provisioned nodes)
        - Exchange pieces of the system (e.g., K8S for OpenStack)
    - Discover and configure the fabric
        - PoC script to bootstrap the fabric at install time (QA)
        - Fabric configuration based on XOS models
    - Support DB migration

Container Orchestration

- Deploy CORD platform components using k8s
    - OpenStack
    - XOS
    - ONOS/ONOS Apps
- Support Container-based VNFs using k8s
    - Common overlay network between OpenStack VMs and Docker Containers
    - Hybrid VM-Container service platform
- Demonstrate Light-and-Right CORD configuration
    - Run a configuration with k8s but no OpenStack

XOS

- Improved Support for Developers

    - Static Analysis for services and manifests
    - Simple Synchronizer template, with most code auto-generated
    - Unit test framework for new Model Policies & Sync Steps
    - Dynamic service on-boarding
- Improved Synchronizer and API performance
    - Auto-generated test coverage to include end-to-end tests
    - Improved Support for Deployment Engineers
    - Better contextual tracing and debugging support
    - Dynamic service on-boarding
    - Enforce Interface models between Service Instances
    - Cleaner visualization of the service graph

QA

- Expand test coverage to include M-CORD and E-CORD
- Extend automated tests to more fully exercise the platform
    - Functional regression tests – Black box tests to make sure base components have not regressed
    - End to end CI/CD tests – To make sure a system can be built from scratch, deployed, and can pass a baseline of tests for both control and for traffic.
    - Performance tests – So can track performance over time
    - Build out performance automation framework
    - Populate framework with a few baseline performance tests