User Scenario 4: Kubernetes

Return to User Experiences

Overview

This topic is a comprehensive performance evaluation of different Kubernetes CNI plugins for Edge-Based and Containerized Publish/Subscribe applications. First we have preliminary of Kubernetes and its Networking Model. Then I will use a real world example to illustrate the feasibility & challenges of deploying containerized DDS application in a Kubernetes Cluster. We then did systematic experiments to learn the DDS performance in such kind of deployments. Based on the observations we can get some conclusions and insights.

Quick Kubernetes Overview

To fully understand the Kubernetes based service its always good practice to have a brief overview of Kubernetes architecture. A Kubernetes Cluster is composed of a Control Plane/Master Node that leads to at least one worker node. A Node of a Kubernetes Cluster could be a virtual machine or physical machine. A Container is nothing but a process that isolates SIG group and namespace from other processes running on the host. However Kubernetes builds another layer of encapsulation on top of containers called a Pod, or a group of 1 or more containers. From the perspective of networking namespace, each node in the Kubernetes Cluster has an individual seeder and any Pod running on the node has a global unique IP. All containers in the Pod share the same network namespace.

Motivation for the Project

Benefits of Deploying DDS Applications In a Kubernetes Cluster

There are several benefits for deploying on Kubernetes such as:

- The container enables lightweight encapsulation of functional modules, such as DDS' publisher/subscriber capabilities
- Offers resource isolation thereby allowing distributed applications and services to be easily deployed and operated across heterogeneous platforms through sharing standardized container images.
- With Kubernetes, it is easy to deploy, update, scale, and self-heal distributed applications.

Kubernetes Networking Model

Since DDS is basically a networking application the Kubernetes network model is very much relavent to DDS' performance when running in containers. Kubernetes network model supports plugable container interfaces, but these plugins have to satisfy the following requirements:

- All containers must communicate with all other containers without Network Address Translation(NAT)
- All nodes must communicate with all containers (and vice-versa) without NAT
- The IP that a container sees itself is the same IP that others see.

There are four types of Networking Schemas:

1. Container-to-Container Networking
2. Pod-to-Pod Networking
3. Pod-to-Service Networking
4. Internet-to-Service Networking

But we will only focus on Pod-to-Pod & Pod-to-Service Networking

Kubernetes Networking Approaches

In this project we studied three virtual network plugins including Flannel, WeaveNet, & Kube-Router. The reason we selected them they are mainstream CNIs that support common multi-platform Edge-to-Cloud deployment.

- Host-Mode networking approach to have a base-line for understanding the overhead of virtual network plugins. When using the host-mode networking approach containers share the same network namespace as the host, so there is no network virtualization so applications can obtain the best performance.

The problem is there may be many port conflicts or security issues because any container has a chance to manipulate the network stack of other containers running on the same host, or containers running on the same host may need the same port number.

Virtual Network plugins can be classified into two groups; Including Layer 2-(Data Link Layer???Needs Check-9:44) and Layer 3-Network Layer implementations.

- Flannel supports both of them with VXLAN & Host-GateWay Backends respectively. VXLAN backend, a Layer-2 implementation, Flannel encapsulates packets with VXLAN header in Linux Kernel using VXLAN tunneling technique. Host-GateWay, a Layer-3 implementation, uses peer hosts as gateways, writing routing rules between source and target pods directly into the host's IPtables.
- WeaveNet utilizes VXLAN's tunneling technique as well but it enables this process with an open vSwitch maintained by a WeaveNet Daemon process.
- Kube-Router, another Layer-3 implementation, uses the iBGP to exchange routing and reachability information when performing inter-subnet communication, while BGP is used instead for cross-subnet networking.

Quantative Performance Evaluation

Multi-Platform Kubernetes Deployment

Why does the hybrid-architecture deployment?

- Edge-to-Cloud continuum is a common deployment schema for managing large-scale IoT apps.
- Kubernetes Master Node is much more resource-intensive than worker nodes due to extensive cluster-wide operations, such as managing cluster state and processing user requests received from its API Server
- Deploying Kubernetes Master Node on resource-constraint edge devices may lead to un-predictive response times especially in large-scale clusters.

Figure 1: Due to the amount of processing required to managed the cluster state, as well as process user interaction and requests from the API servers. The server with more processing power was able to outperform the more constrained server.

Benchmark Framework

Next we will show our Benchmark Framework to illustrate the feasibility of running DDS container in a Kubernetes Cluster. In this framework there are several components;

- Firstly the Auto Deployer works on setting up the whole cluster and installing the same plugins on the worker and Master node as well as deploy the Perftest container on worker nodes.
- Perftest Pod: a containerized Perftest benchmark application compiled with Connext DDS
- Since some Kubernetes CNIs don't support Multicast so we deployed DDS Cloud discovery service as a Kubernetes deployment to enable service discovery between publisher and subscribers
- The Resource Metrics Monitor is Kubernetes built-in metric server for record resource utilization for node and pod.
- We also used DDS-Perftest Manager that generates performance test commands with a given experiment profile, then executes them on pods and monitors the experiment's progress & record performance metrics.

Experiment Environment

Measurement Approach

1. Throughput
    - 1-sub Scenario: Publisher sends data samples at a configurable rate, and a subscriber calculates throughput by counting the volume of received bytes and samples
    - N-sub Scenario: Average throughput over all subscribers
2. Latency
    - 1-sub Scenario: The publisher sends a “ping” to the subscriber and enters into a waiting state. The subscriber receives the ping and responds with a “pong” and the Publisher will then determine the one way latency from the measured roundtrip time.
    - N-sub Scenario:

Figure is Temporary

Experiment 1Virtualization Overhead

The purpose of the first experiment is to understand the overhead of container virtualization. We setup the experiment with 1 publisher and 1 sub and the publication range is unlimited for throughput test. We enabled the reliability QoS but disabled the micro batching QoS Should be noted that BareMetal means we ran the Perftest application on the Baremetal machine, But for host-mode we deployed the Perftest container but the container used host network namespace. So in this case we can observe the overhead of container virtualization. In our observations showed that the impact of container virtualization does not impact the real time properties of DDS application. From the pie charts you can see the overhead of either throughput and latency is trivial.

Experiment 2 CNI Overhead

Our second experiment is to learn the performance overhead illistrated by Virtual network plugins, similarily we did 1pub-1sub experiments with similar QoS Settings. We found that Flannel-VXLAN and WeaveNet present notable throughput degradation over all payload lengths due to the overhead of L2 encapsulation WeaveNet's throughput is comparably lower than Flannel-VXLAN, suggesting Open vSwitch VXLAN encapsulation introduce more overhead. Flannel-Hostgw and Kube-Router obtain similar throughput because they are both Layer-3 But Flannel-Hostgw performs better than kube-router when the payload length is grater than 1KB due to the difference in MTU(1500B vs 1480B)