가상 네트워크와 컨테이너 인터페이스 기반 오픈 클라우드 컴퓨팅 플랫폼 연구

김기현*, 김동균**, 김용환*
*한국과학기술정보연구원
**한국과학기술정보연구원
e-mail:{kkh1258, mirr, yh.kim086}@kisti.re.kr

Research on Open Cloud Computing Platform Based on Virtual Network and Container Interface

Ki-Hyeon Kim*, Dongkyun Kim**, Yong-Hwan Kim*
*Korea Institute of Science and Technology Information
**Korea Institute of Science and Technology Information

요 약

데이터 센터를 기반으로 서비스를 수행하는 기업들은 비용절감을 위해 서버 가상화 기술을 이용한다. 서버 가상화를 이용하는 기업들은 대부분 하이퍼바이저 기반의 서버 가상화 기술을 사용하며, 이 경우 하드웨어 가상화를 통해 커널 단에서 많은 I/O와 리소스를 처리해야 한다. 따라서 하이퍼바이저 기반의 서비스는 느리다는 단점이 있으며 이를 해결하기 위해 컨테이너 기반의 가상화 기술을 이용할 수 있다. 하지만 컨테이너 기반의 네트워크 또한 문제점이 존재한다. 컨테이너 기반의 네트워크는 유연한 네트워크를 구성하기 어렵고, 기존의 컨테이너 네트워크 인터페이스를 활용할 경우 데이터 전송 성능이 저하된다. 본 논문에서는 컨테이너 오케스트레이션 툴인 Kubernetes와 SDN (Software-Defined Network) 기반의 가상전용 네트워크 연계 환경을 구축하고 이에 적합한 컨테이너 네트워크를 연구하여 이의 문제점을 해결한다. 즉, 가상전용 네트워크와 Kubernetes의 연계를 통해 고성능의 유연한 네트워크를 구성할 수 있는 프레임워크를 개발하여 기존 컨테이너 기반 네트워크와 비교하고 성능을 검증했다.

1. 서론

데이터 센터를 기반으로 서비스를 운영하는 Amazon, Google, Naver, 다음카카오 등 많은 기업들은 데이터센터를 기반으로 서비스를 수행한다. 이와 같은 기업들은 데이터센터 운영의 비용을 줄이기 위해 노력한다. 서버들은 평균 15% 미만으로 매우 낮은 활용도를 보여주며, 가상화 소프트웨어는 그러한 활용도를 평균 4배 이상 높여준다. 이는 가상화될 수 있는 모든 워크로드의 경우에 기업들이 물리적 서버의 수를 1/4정도로 줄일 수 있다는 것을 의미한다[1]. 데이터센터 측면에서 비용을 줄이기 위해서는 서버 자원의 낭비를 줄여야 한다. 서버 자원의 낭비를 줄이기 위해 기업들은 서버 가상화 기술을 사용하는데, 서버 가상화 기술은 서버의 리소스들을 논리적으로 나누어 다양한 서비스를 수행할 수 있는 기술을 말한다. 서버 가상화 기술을 이용하는 방법은 하이퍼바이저를 이용한 기술과 컨테이너를 이용한 기술로 나눌 수 있다.

하이퍼바이저를 이용한 기술의 경우 OS를 자유롭게 설치할 수 있는 반면 하드웨어 가상화를 사용하기 때문에 커널 단에서 많은 I/O를 처리해야 한다. 이에 따라 서비스를 수행할 시 시스템이 무겁고 느리다는 문제점이 있다[2]. 이와 같은 문제점을 해결하기 위해 컨테이너 기반의 가상화 기술이 등장하였다. 컨테이너는 가상화 기술의 하나이며, 추가적인 OS를 설치하여 가상화하는 방법을 개선하기 위해 프로세스를 격리하여 가볍고 빠르게 동작할 수 있는 기술이다. 컨테이너 기반의 오픈소스 가상화 플랫폼을 Docker라 하며, 구글에서 Docker를 오케스트레이션하는 툴인 Kubernetes[3]을 개발했다. Kubernetes는 여러 대의 호스트에서 컨테이너를 실행시켜주며, 호스트 간 컨테이너 네트워킹과 호스트 머신의 리소스를 분배 등의 문제를 해결해 줄 수 있는 툴이다. 다만 컨테이너 네트워크 인터페이스(Container Network interface, CNI)를 구성할 때, 기존 CNI의 경우 성능 저하 문제가 제기되며, 네트워크의 유연성을 보장하지 못한다는 문제점이 존재한다[4].

KREONET[5]은 한국과학기술정보연구원 (KISTI)이 관리· 운영하는 국가 R&D 연구망이다. KISTI에서는 KREONET 기반의 차세대 서비스 모델로 소프트웨어 정의 네트워크(SDN) 기반의 광역통신망 인프라인 KREONET-S[6]를 구축 및 개발하고 있다. KREONET-S는 국내 및 국제 네트워크를 포함하고 있으며, KREONET-S 인프라의 모든 네트워크 요소는 새로운

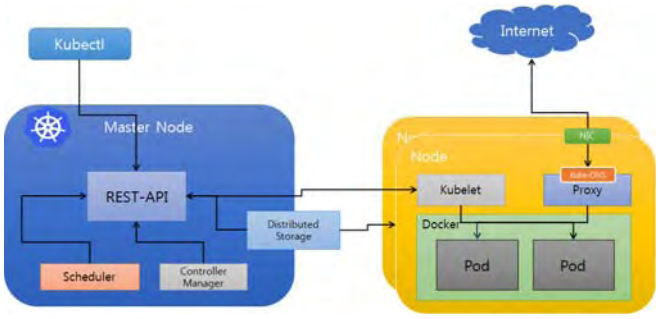
* 본 연구는 2018년도 한국과학기술정보연구원(KISTI) 주요사업 과제로 수행한 것입니다.

SDN 네트워크 운영, 관리 및 서비스를 위한 Open Network Operating System (ONOS)[7] 제어 플랫폼에 의해 동작한다. 특히 KREONET-S는 사용자들이 요구하는 고성능의 네트워크를 짧은 시간 내 구축하여 대용량 데이터 전송과 관리를 요구하는 첨단협업연구를 수행할 수 있도록 가상전용 네트워크 (Virtually Dedicated Network, VDN)의 동적 구축 서비스를 제공한다. VDN의 주요 서비스로는 VDN 슬라이싱, VDN 페더레이션, 가상망 접근제어(vNAC), VDN DHCP 등이 있다. VDN 슬라이싱은 오픈 플로우의 미터 테이블을 활용하여 임의의 호스트 그룹에 대해 사용자가 요구하는 대역폭을 보장할 수 있는 네트워크를 제공하는 기능이다. VDN 페더레이션은 서로 다른 SDN 네트워크 도메인 간 연계 정보를 생성하여 JSON 형식으로 교환하고, 이를 기반으로 상호간 연결 호용된 호스트들 사이의 통신을 가능하게 하는 기능이다. vNAC은 외부 IP 게이트웨이와 연결된 호스트들이 외부 네트워크와의 데이터 통신 보안을 강화할 수 있는 기능이고, vDHCP는 가상전용망 참여 호스트들의 IP를 자동으로 할당받을 수 있도록 설정하는 기능이다.

본 논문에서는 대용량의 과학데이터의 고속 전송 서비스를 사용자들에게 제공할 수 있도록 VDN과 Kubernetes의 연계를 통해 새로운 CNI를 개발하고 성능 시험을 수행했다. 개발된 CNI를 KREONET-S에 적용하여 ONOS 컨트롤러를 통한 시스템 가시화를 진행했으며, 성능 시험을 통해 해당 CNI의 우수성을 보이고자 한다.

2. 오픈 클라우드 컴퓨팅 플랫폼 개발

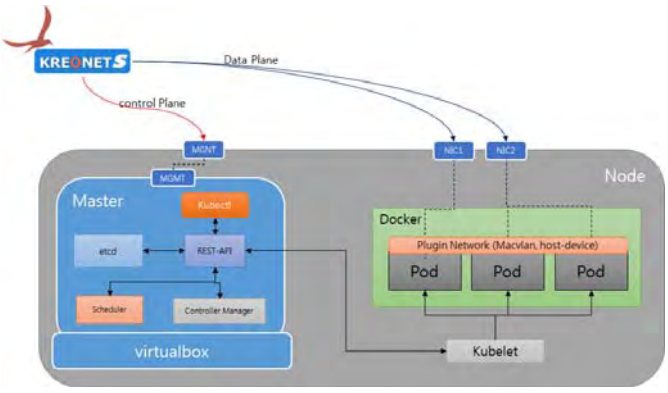
본 논문에서는 VDN과 Kubernetes의 연계를 통하여 오픈 클라우드 컴퓨팅 플랫폼을 개발하였다. 기존의 Kubernetes의 네트워크 구조와 개발한 오픈 클라우드 컴퓨팅 플랫폼에서 Kubernetes의 네트워크 구조의 차이를 설명하며, 기존에 사용하던 CNI와 개발한 CNI의 차이에 대해 설명한다.



(그림 1) Kubernetes의 구조

기존의 Kuberntes의 구조는 그림 1과 같다. 기존의 Kubernetes의 네트워크 구조는 Proxy를 이용하여 주요한 네트워크 룰을 설정하고 로드밸런싱을 수행한다. Kubernetes에서 생성되는 Pod들은 처음 생성될 때 IP를 배정받아야 생성되기 때문에 DHCP 서버가 필요하며, 생성될 때 어떤 Node에 생성될지 모르기 때문에 리소스의 위치 정보가 필요하다. 이와 같은 네트워크 패턴을 Service Discovery Patten이라고 한다. Kubernetes는 Service Discovery Patten을 해결하기 위해 내부 DNS서버를 이용한다. 새로운 리소스가 생성되면, 리소스에 대한 IP와 DNS이름을 맵핑하여 리소스에 접근할 수 있도록 구성되어 있다.

본 논문에서 개발한 오픈 클라우드 컴퓨팅 플랫폼의 구조는 그림 2와 같다. 오픈 클라우드 컴퓨팅 플랫폼에서 Kubernetes의 네트워크는 VDN과의 연계를 통하여 네트워크를 구성한다. Kubernetes에서 리소스를 생성하기 위해서는 리소스 생성 이전에 IP를 할당 받아야한다. IP를 할당받기 위해서는 기존에 네트워크를 수행하던 Proxy의 기능을 제거하고, VDN의 Rest-API를 이용하여 vDHCP로 부터 IP를 할당 받는다. Kubernetes의 마스터 클러스터에서 컨트롤플레인을 통해 VDN REST-API를 이용하여 vDHCP 정보를 가져와 Pod에 IP를 할당한다. 생성된 Pod은 개발된 CNI 두 가지 중 하나를 사용하여 데이터플레인을 구성 후 Pod의 통신을 수행할 수 있다.



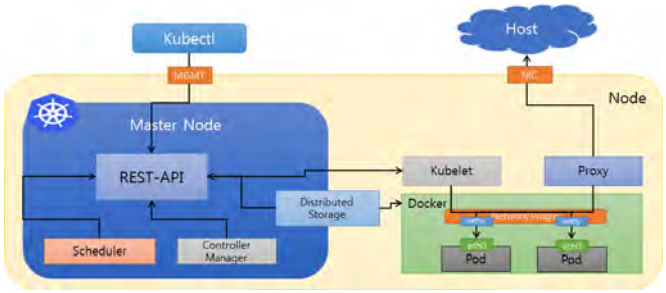
(그림 2) 오픈 클라우드 컴퓨팅 플랫폼 구조

본 논문에서 개발한 CNI는 Macvlan과 Host Device 두 가지의 네트워크이다. 기존에 Kubernetes에서 사용하는 CNI들은 브리지 네트워크와 오버레이네트워크 방식을 사용하였기 때문에 성능 저하의 문제가 생겼지만, 본 논문에서 개발한 CNI는 컨테이너와 호스트 사이의 브리지를 제거함으로써 보다 간단하고 원활한 네트워크의 구조로 개발되었다. 또한 두 가지의 기능의 차이점은 Macvlan은 다수의 호스트를 연결할 수 있도록 구성하기 위하여 개발되었으며, Host Device의 경우 하나의 Pod이 하나의 NIC을 할당받아 네트워크를 구성하기 때문에 높은 대역폭을 필

요로 하는 시스템에 적용하기 위하여 개발되었다.

3. Kubernetes 성능 테스트 환경

본 장에서는 기존에 사용하던 CNI와 개발된 CNI를 적용한 Kubernetes를 이용하여 네트워크를 구성 후 네트워크의 성능 테스트를 수행하고자 한다.



(그림 3) Kubernetes 성능 테스트 환경

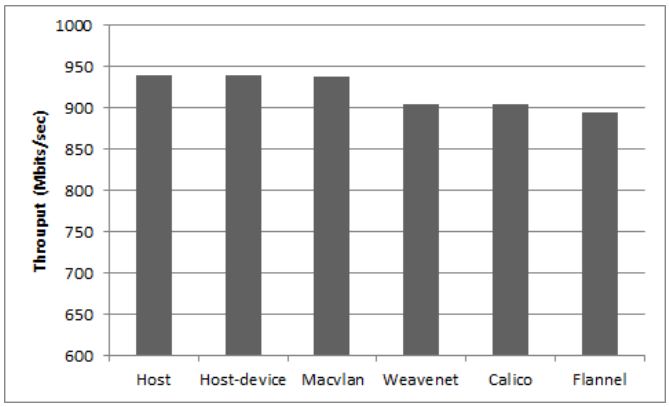
Kubernetes의 CNI 성능 테스트를 수행하기 위한 환경은 그림 3과 같다. 한 대의 서버에 Master와 Node를 설치하여 Kubernetes를 구성하였으며, Master의 경우 Virtualbox를 이용하여 가상머신(VM) 위에 Master를 구성하였으며, Node는 실제 서버 호스트에 구성하였다. LAN 구간이 아닌 WAN 구간에서의 CNI 성능 검증을 위하여 대전에 그림 3의 Kubernetes 환경을 구성하고 부산에는 성능 측정 서버를 구축하였다. 그리고 SDN을 기반으로 대전의 Kubernetes로 생성한 Pod과 부산의 성능 측정 서버 사이에 1Gpbs 대역폭을 지닌 가상망을 구성하여 성능 테스트를 진행하였다. 이의 실험에서는 기존에 사용하던 CNI 중 Calico, Flannel, Weave Net을 사용하였으며, 본 논문에서 개발한 CNI 또한 동일한 환경에서 성능 테스트를 수행하였다. 또한 개발된 CNI를 적용한 Kubernetes 환경간의 성능 테스트를 위하여 대전과 부산에 각각 독립적인 Kubernetes 테스트베드 환경을 구성하고, 이들 사이의 10Gbps 가상망을 생성하여 대전의 Pod과 부산의 Pod 간에 성능 측정을 수행하였다.

한편, 실험의 데이터를 수집하기 위해 서버의 네트워크 성능 측정 프로그램인 Iperf3를 이용하여 성능 측정을 수행하였으며, 각 CNI 별로 1000초 동안의 성능측정을 수행하고 이의 데이터를 수집하였다.

4. 성능 결과

본 장에서는 Kubernetes에서 사용되고 있는 기존의 CNI인 Calico, Flannel, Weave Net과 본 논문에서 개발한 CNI의 성능 결과를 보인다.

표 1은 Kubernetes의 5 가지 CNI의 성능 결과를 나타낸 표이다. 그림 4은 1Gbps 테스트베드를 통하여 호스트와 성능측정 서버와의 성능측정 결과와 기존의 Kubernetes의 3 가지 CNI의 성능 측정 결과와 개발된 CNI의 성능측정 결과 그래프이다. 그림 4를 보면 Calico는 평균 904Mbits/sec의 성능을 보이며, Flannel의 경우 평균 895Mbits/sec의 성능을 보이며, Weave Net의 경우 평균 905Mbits/sec의 성능을 보인다. 한편, 본 논문에서 개발한 CNI인 Macvlan의 성능은 938Mbits/sec의 성능을 보이며, Host device의 경우 평균 940Mbits/sec의 성능을 보인다. 즉, 1Gbps의 대역폭에서 성능측정을 수행하였을 때, Macvlan과 Host device의 성능이 실제 호스트의 성능과 유사한 성능을 보이며, 기존의 CNI 3가지에 비해 30Mbits/sec 이상 성능이 높다는 것을 알 수 있다.

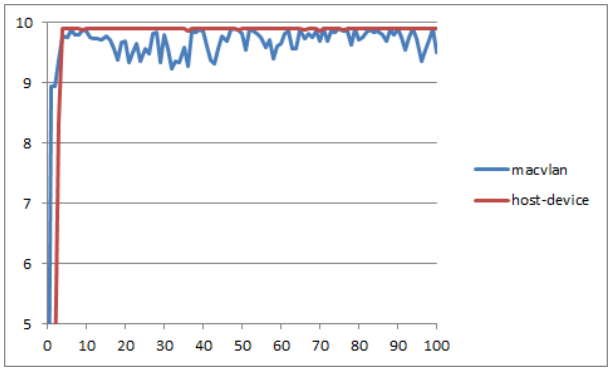


(그림 4) Kubernetes 1Gbps 테스트베드를 통한 CNI 성능 측정 결과

10Gbps의 대역폭에서 개발된 CNI를 적용하여 실험을 수행하였을 때, Macvlan의 경우 평균 9.7Gbits/sec의 성능을 보였으며, Host device의 경우 평균 9.88Gbits/sec의 성능을 보였다. 그림 5의 그래프를 보면 Macvlan의 경우 성능은 높지만, 불안정한 네트워크 성능을 보인다. 하지만 Host-device의 경우 안정적인 네트워크 성능을 확인할 수 있다. 이는 Host-device의 경우 실제 호스트의 NIC을 할당받아 사용하기 때문에 높고 안정적인 성능을 보장 할 수 있다. 또한 개발된 CNI의 경우 VDN과 연계하여 유연한 네트워크를 사용할 수 있다. VDN의 네트워크 가상화 기능을 이용하여 기존의 네트워크와 같이 복잡한 네트워크 설정 없이 사용자들의 요구사항에 적절한 네트워크를 제공할 수 있으며, 보안에 강화된 네트워크를 구성할 수 있다.

그림 6은 오픈 클라우드 컴퓨팅 플랫폼을 KREONET-S에 적용하였을 때, Kubernetes에서 생성된 Pod이 ONOS 컨트롤러를 통하여 시스템 가시화의 가능성을 확인한 그림이다. 그림을 통해 대전과 부산에 설치된

Kubernetes 시스템을 통해 Macvlan과 Host Device CNI를 이용하여 Pod을 생성하였고, 현재 생성된 Pod이 ONOS UI에서 확인이 가능한 것을 확인하였다.



(그림 5) Kubernetes 10Gbps 테스트베드를 통한 개발된 CNI 성능 측정 결과

(그림 6) ONOS GUI를 통해 연결된 Pod 확인

5. 결론

데이터 센터에서 가볍고 확장성 높은 서비스를 제공하기 위해 컨테이너 기반의 서버 가상화 기술을 사용한다. 컨테이너 기반의 서버 가상화의 문제점으로 네트워크의 성능 저하 및 유연성의 문제점이 제기 된다. 이를 해결하기 위해 Kubernetes와 VDN을 연계한 오픈 클라우드 컴퓨팅 플랫폼을 개발하고 있다. 본 논문에서는 오픈 클라우드 컴퓨팅 플랫폼에서 개발된 CNI와 기존의 Kubernetes에서 사용되던 CNI인 Calico, Flannel, Weave Net, Macvlan, Host Device을 Software-Define Wide Area Network (SD-WAN) 구간인 대전과 부산 구간에 시스템을 구성하여 네트워크 성능 측정을 수행한다. 1Gbps 대역폭에서 테스트를 수행한 성능 측정 결과 Calico의 경우 904Mbis/sec, Flannel의 경우 895Mbits/sec, Weave Net의 경우 905Mbits/sec의 성능 결과를 보였으며, 개발한 CNI 중 Macvlan의 경우 938Mbits/sec의 성능을 보였으며, Host Device의 경우 940Mbits/sec의 성능을 보였다. 성능 결과 기존의 CNI를 사용할 경우 브릿지와 오버레이 방식을 이용하여 네트워크를 구성하여 높은 성능 결과를 가져오기 힘들었지만, 개발한 CNI의 성능은 실제 호스트의 성능과 유사한 성능 결과를 가져왔다. 또한 10Gpbs 대역폭에서 테스트를 수행한 결과 9.7Gbits/sec 이상의 성능 결과를 보였다. 이를 통해 개발된 CNI의 성능 우수성을 증명하였다.

6. 향후 연구 계획

본 논문에서 개발한 오픈 클라우드 컴퓨팅 플랫폼을 이용하여 가상화 환경에서 고성능의 네트워크를 구성할 수 있었다. 추후에는 본 논문에서 개발한 플랫폼을 이용하여 고성능 과학기술 빅데이터 전송 노드인 Data Transfer Node (DTN)[8]를 구성하고, KREONET-S를 통해 동적 네트워킹을 구성할 수 있는 Science DMZ[9] 모델의 가상화를 연구할 예정이다.

참고문헌

[1] Gartner, “Identifies 10 Key Actions to Reduce IT Infrastructure and Operations Costs by as Much as 25 Percent”, Sep., 28, 2011

[2] Roger S. Pressman "Software Engineering A Practltiners' Approach" 3rd Ed. McGraw Hill

[3] Felter, Wes, Alexandre Ferreira, Ram Rajamony, and Juan Rubio. "An updated performance comparison of virtual machines and linux containers." In Performance Analysis of Systems and Software (ISPASS), 2015, pp. 171-172.

[4] Dusia, Ayush, Yang Yang, and Michela Taufer. "Network quality of service in docker containers." In Cluster Computing (CLUSTER), 2015, pp. 527-528.

[5] KREONE web site, Retrieved Sep., 06, 2018, from http://www.kreonet.net/

[6] KREONET-S web site, Retrieved Sep., 08, 2018, from http://www.kreonet-s.net/

[7] ONOS web site, Retrieved Sep., 07, 2018, from http://www.onosproject.org/

[8] ESnet web site, Retrieved May., 22, 2018, from https://es.net/science-dmz/DTN/

[9] Dart, Eli, Lauren Rotman, Brian Tierney, Mary Hester, and Jason Zurawski. "The science dmz: A network design pattern for data-intensive science." Scientific Programming 22, no. 2 (2014): 173-185.