GEdge(Griffin-Edge) Platform
- 초저지연 지능형 클라우드 엣지 SW 플랫폼 -

지능형 엣지 서비스 실행 가속을 위한 SW/인프라기술

2021.12.09.

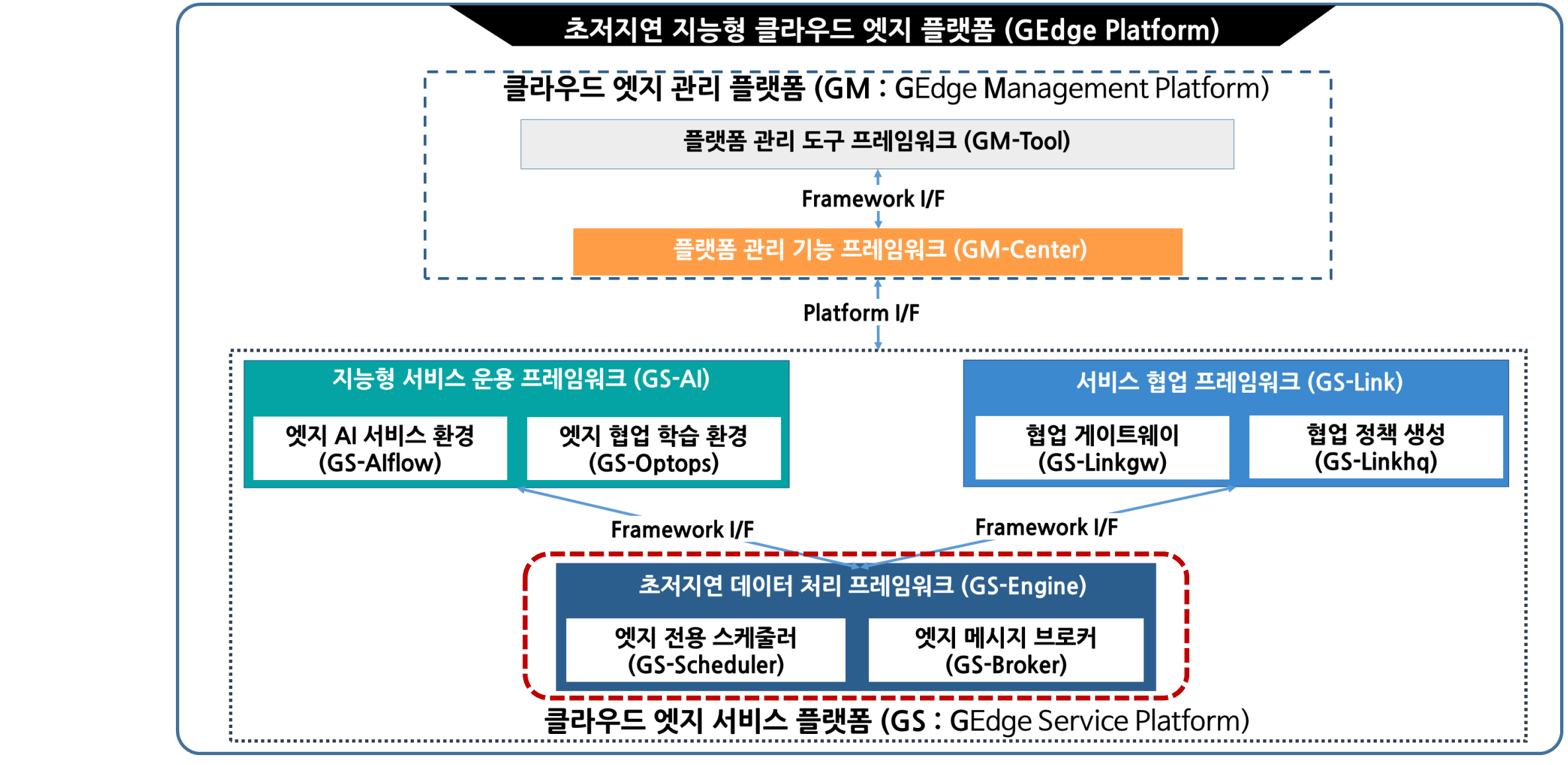
GS-Engine 프레임워크 리더
최현화(hyunwha@etri.re.kr)

"GEdge Platform" 은 클라우드 중심의 엣지 컴퓨팅 플랫폼을 제공하기 위한
핵심 SW 기술 개발 커뮤니티 및 개발 결과물의 코드명입니다.

- Developer-Friendly

GEdge Platform Community 3rd Conference (GEdge Platform v2.0 Release) -

이번 발표의 기술적 포지셔닝



Contents

I 지능형 엣지 서비스(AI)

II GSE API Server

III 기술 데모 및 향후 계획

지능형 엣지 서비스(AI)

1 지능형 엣지 서비스의 영역은?

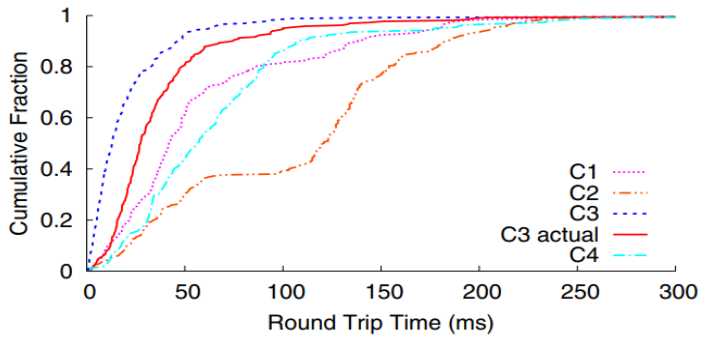
AI can do Everything

Ref: GTC Spring 2021 Keynote with NVIDIA CEO Jensen Huang, https://www.youtube.com/watch?v=eAn_oiZwUXA

2 지능형 엣지 서비스 지원 방법

- 지능형 엣지 서비스 성능 비교

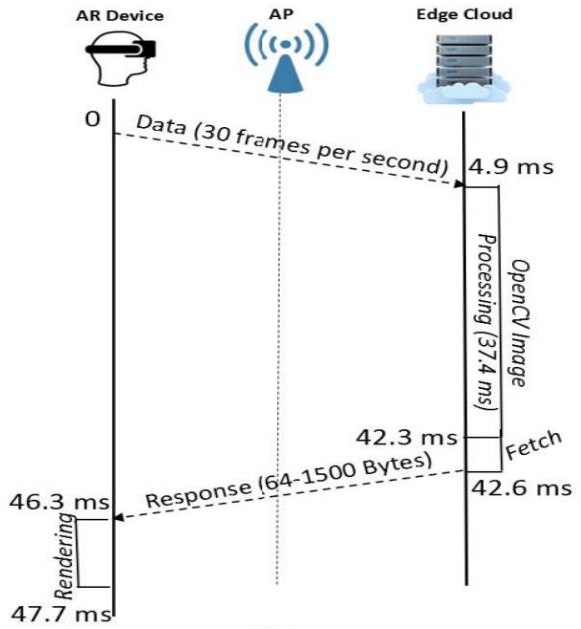
Public Cloud



The latency of services deployed by cloud providers is over 100ms

Ref: CloudCmp: Comparing Public Cloud Providers (ACM IMC 2010)

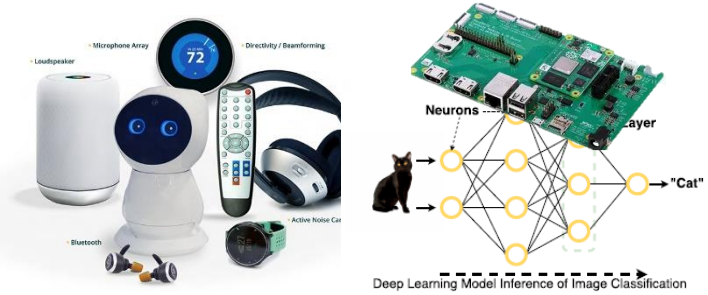
Edge Cloud

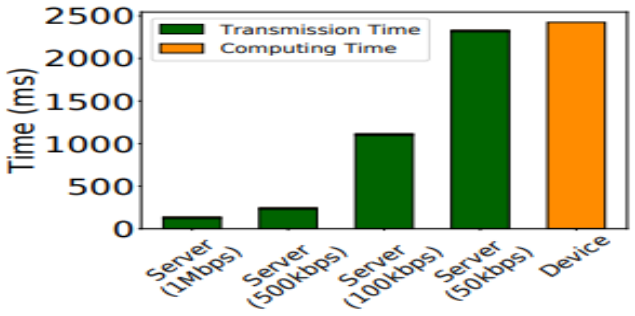


The latency of services has 50ms

Ref: Scalability and Performance Evaluation of Edge Cloud Systems for Latency Constrained Application (ACM/IEEE SEC, 2018)

Smart Device





It takes more than 2s to execute the model on the resource-limited Raspberry PI

Ref: Edge AI: On-Demand Accelerating Deep Neural Network Inference via Edge Computing(IEEE transactions on wireless communications, 2019)

II GSE API Server

1 CPU 자원 선점 (1)

- 지연 시간 민감형 서비스
  - (Target 응용 특징) CPU intensive / IO intensive
  - (성능 강화 방안) Context switching/CPU affinity

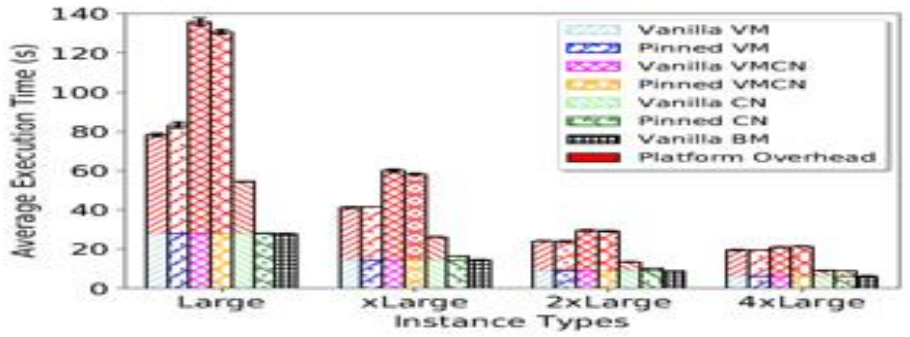


Fig. 3: Comparing execution time of FFmpeg on different execution platforms under varying number of CPU cores.

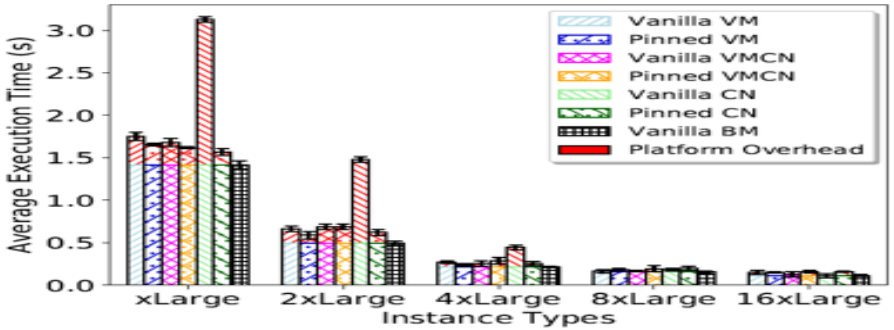
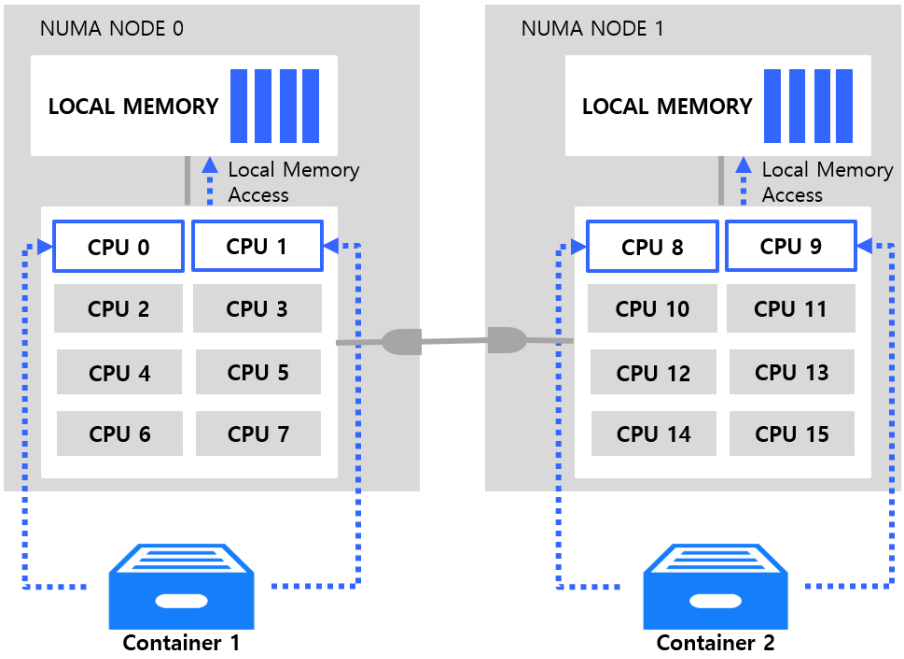


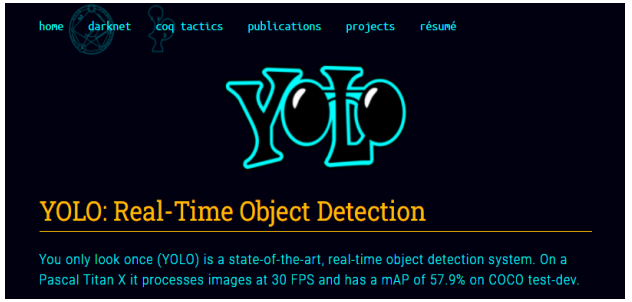
Fig. 5: Comparing mean response time (aka execution time) of 1,000 web processes on different execution platforms (WordPress evaluation)



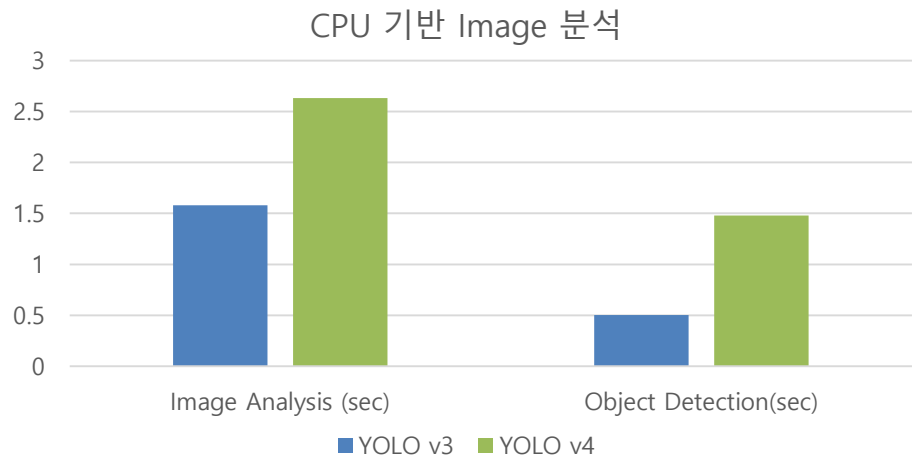
Ref: The Art of CPU-Pinning: Evaluating and Improving the Performance of Virtualization and Containerization Platforms (2006)

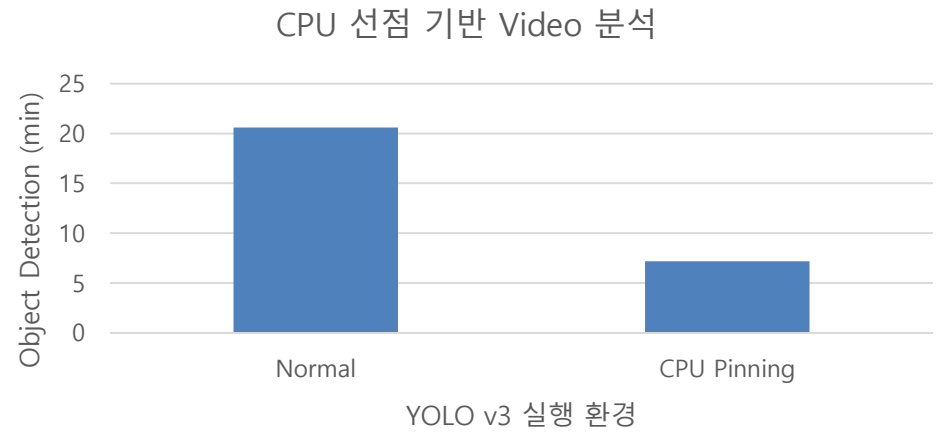
2 CPU 자원 선점 (2)

- CPU 기반 성능 실험
  - 2020년 : CPU 자원 선점 기반으로 동영상 인코딩 (**52% 성능 향상**)
  - 지능형 서비스의 성능 실험: YOLO 기반 객체 인식(v1~v5)
  : 실시간 동영상(스트리밍) 분석 지원 **(300% 성능 향상)**
  : (1 FPS) 쿠팡 물류센터 화재/침입탐지형 서비스는 CPU만 분석 가능



https://pjreddie.com/darknet/yolo/





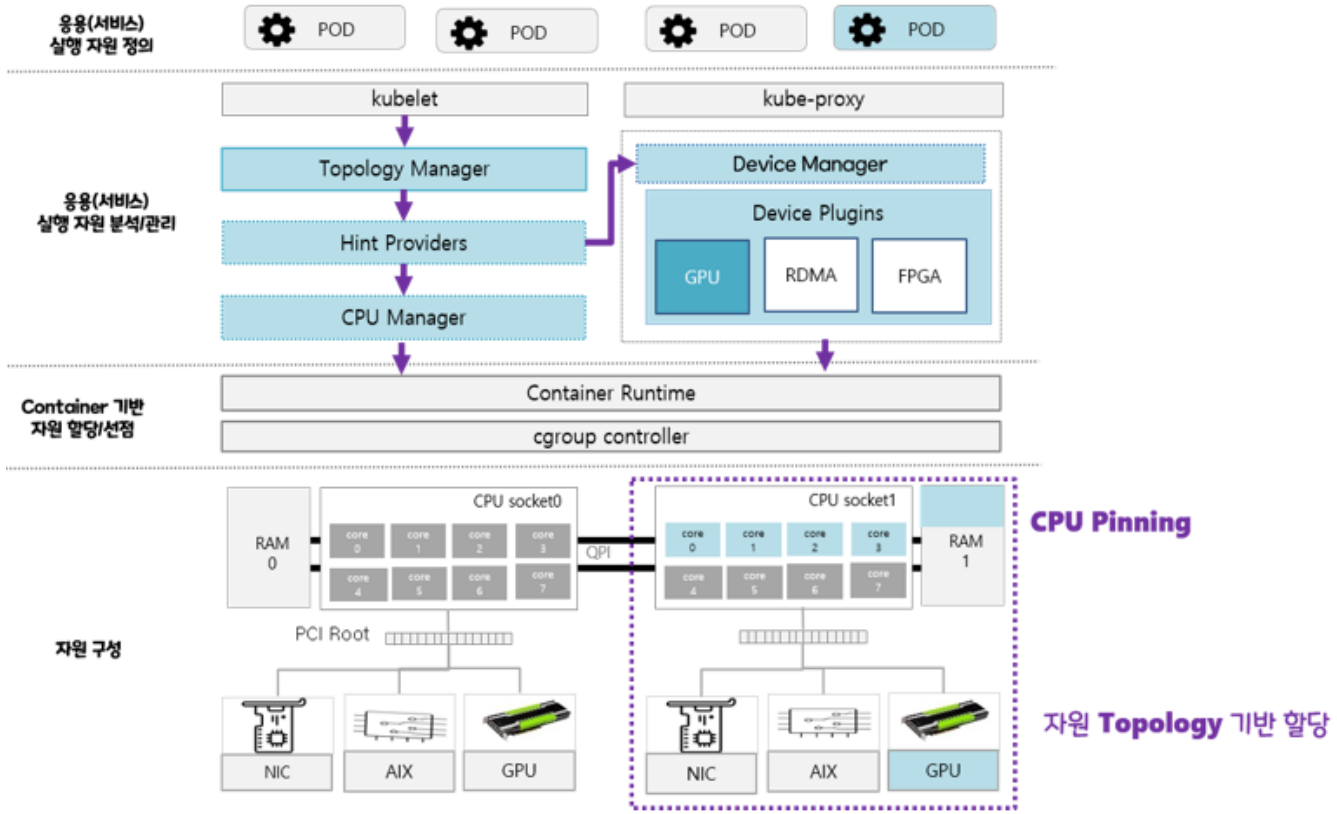
- YOLO v3 우수: v3(107 layer), v4(162 layer)
  => Precision은 거의 동일, v3 41% 빠름
- 이미지 분석 시간에서 Object detection 은 33% 차지

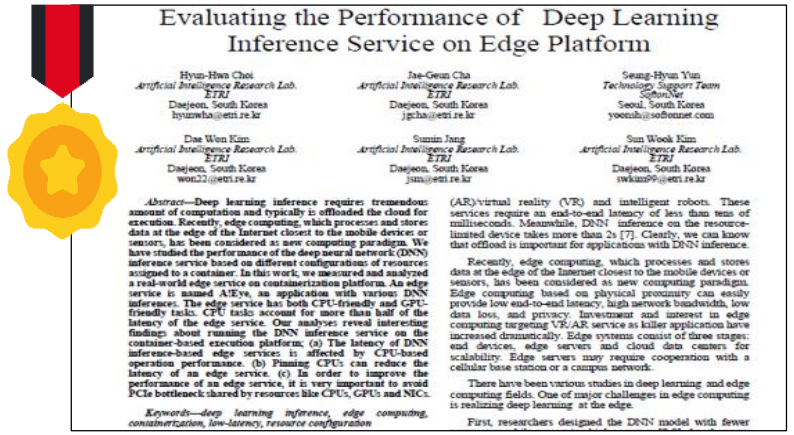
- YOLO v3 기반 Scooter.mp4 분석
  => HD(1280*720), 14sec(30.481KB)
- 300% 성능 향상 (1 FPS 분석)
  => Sliding Window 분석 속도

Ref: You Only Look Once: Unified, Real-Time Object Detection (2016)

3 토폴로지 인지형 자원 선점 (1)

- 초저지연 지능형 서비스 지원
  - 자원간 Affinity / Data copy 최소화
  - **CCTV와 같은 스트리밍 분석 (CPU/GPU tasks)**





Evaluating the Performance of Deep Learning Inference Service on Edge Platform

Hyun-Hwa Choi, Jae-Geun Cha, Seung-Hyun Yun, Dae Won Kim, Sumin Jang, Sun Wook Kim
Artificial Intelligence Research Lab. ETRI, Daejeon, South Korea

Abstract—Deep learning inference requires tremendous amount of computation and typically is offloaded the cloud for execution. Recently, edge computing, which processes and stores data at the edge of the Internet closest to the mobile devices or sensors, has been considered as new computing paradigm. We have studied the performance of the deep neural network (DNN) inference service based on different configurations of resources assigned to a container. In this work, we measured and analyzed a real-world edge service on containerization platform. An edge service is named AIEye, an application with various DNN inferences. The edge service has both CPU-friendly and GPU-friendly tasks. CPU tasks account for more than half of the latency of the edge service. Our analyses reveal interesting findings: about running the DNN inference service on the container-based execution platform; (a) The latency of DNN inference-based edge services is affected by CPU-based operation performance. (b) Pinning CPUs can reduce the latency of an edge service. (c) In order to improve the performance of an edge service, it is very important to avoid PCIe bottleneck shared by resources like CPUs, GPUs and NICs.

Keywords—deep learning inference, edge computing, containerization, low-latency, resource configuration

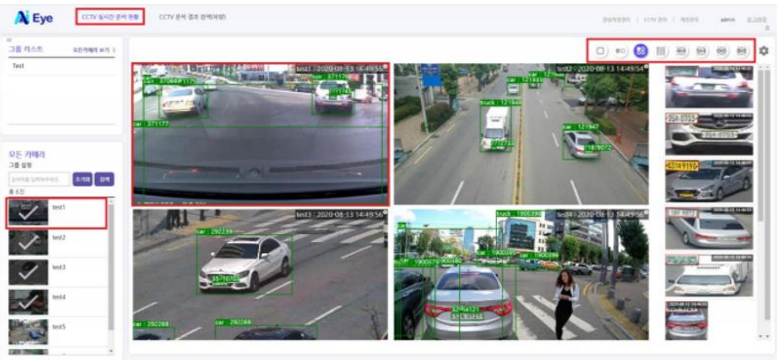
(AR)/virtual reality (VR) and intelligent robots. These services require an end-to-end latency of less than tens of milliseconds. Meanwhile, DNN inference on the resource-limited device takes more than 2s [7]. Clearly, we can know that offload is important for applications with DNN inference.

Recently, edge computing, which processes and stores data at the edge of the Internet closest to the mobile devices or sensors, has been considered as new computing paradigm. Edge computing based on physical proximity can easily provide low end-to-end latency, high network bandwidth, low data loss, and privacy. Investment and interest in edge computing targeting VR/AR service as killer application have increased dramatically. Edge systems consist of three stages: end devices, edge servers and cloud data centers for scalability. Edge servers may require cooperation with a cellular base station or a campus network.

There have been various studies in deep learning and edge computing fields. One of major challenges in edge computing is realizing deep learning at the edge.

First, researchers designed the DNN model with fewer

Vehicle Plate Analysis



SW requirement

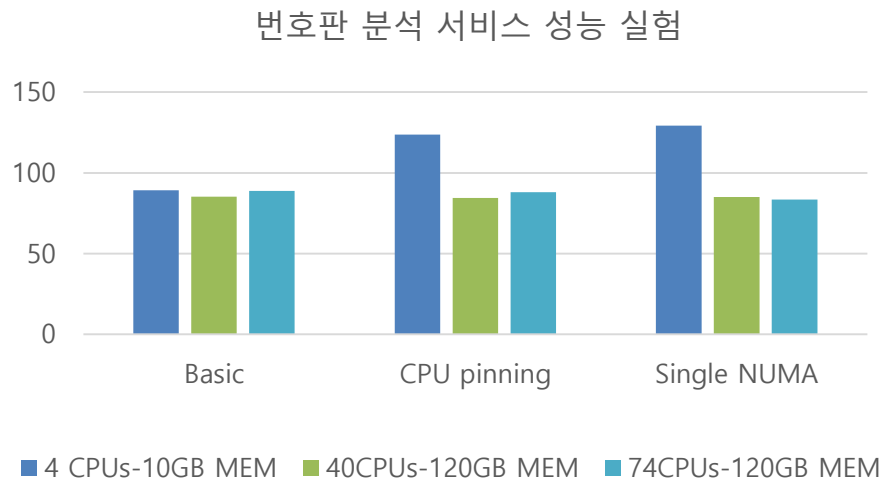
- GPU driver
- CUDA 10.1+
- cuDNN 7.6 +
- Python 3.7 +
- Python-pip 20.2.x +
- Virtualenv 20.0.x +
- TensorRT 6.0
- TensorFlow 2.1

4 토폴로지 인지형 자원 선점 (2)

- 자동차 번호판 인식
  - Data: FHD(1920*1080) 709MB (total frames: 107,836)
  - Model: SSD_MobileNet v2, Lenet-prelu, DeepSORT

성능 실험 결과 (1 container, 1 노드)

	4 CPUs-10GB MEM	40CPUs-120GB MEM	74CPUs-120GB MEM
Basic	89.1	85.24	88.72
CPU pinning	123.71	84.42	88.06
Single NUMA	129.17	84.94	83.47



성능 실험 결과 (2 container, 멀티 노드)

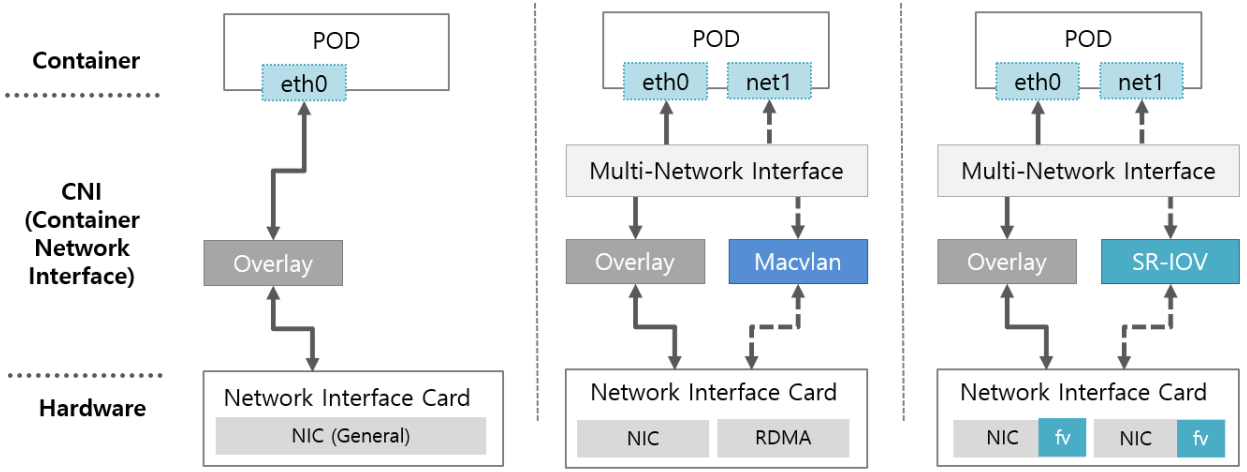
	1Node-1Pod	1Node-2Pods	2Nodes-2Pods	2Nodes-2Pod(/w CPU Pinning)	2Nodes-2Pod(148CPUs)	2Nodes-2Pods(240GB)
analysis time (msec)	88.72	86.57	97.16	97.03	98.9	97.16

- 엣지 서비스 성능은 CPU 작업과 GPU 작업 성능에 의존 (CPU:GPU = 58: 42)
- CPU를 선점시키는 것은 엣지 서비스 성능 향상(3%)

→ 대용량 데이터 입력, 다중 모델(데이터 이동)/이종 자원 → PCIe 공유자원 bottleneck

5 하드웨어 가속 기반 네트워크 이원화

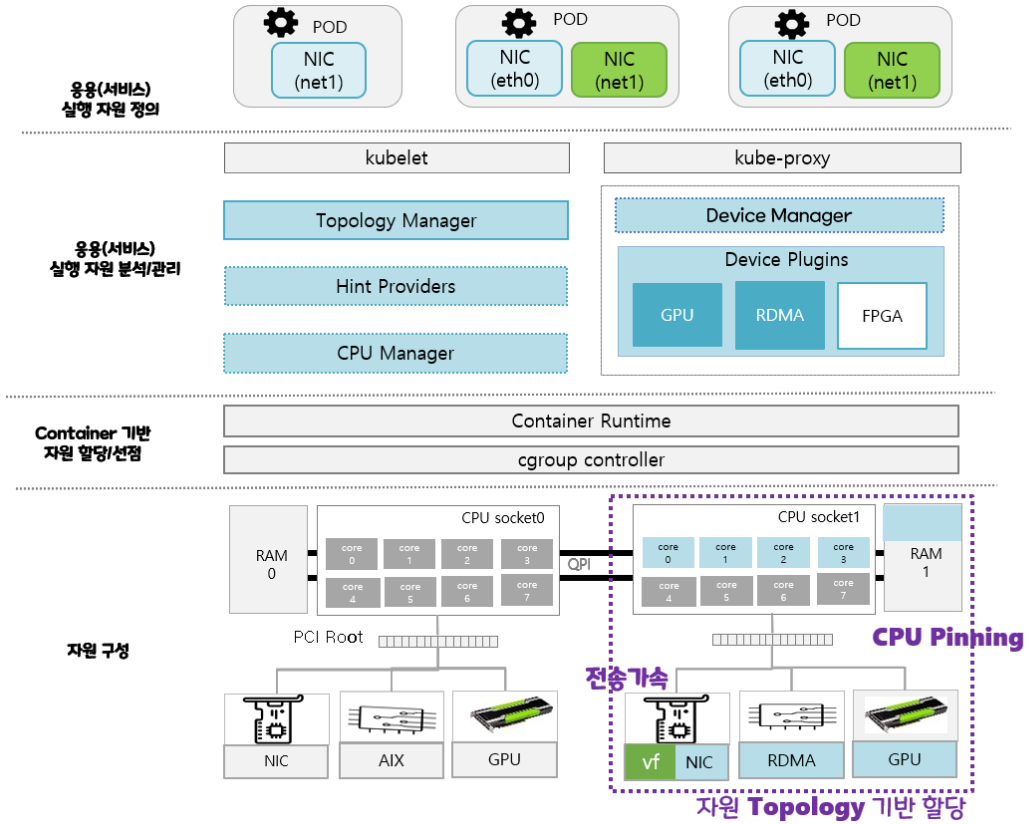
- 응용간 데이터 전송 가속
  - 클러스터 관리 네트워크: Flannel
  - Communication-intensive 서비스 가속
    : 데이터 전송 패스 이원화 : 다중 NIC 컨테이너 지원
    : 데이터 전송 하드웨어 가속 (SR-IOV)
    : RDMA 지원



① 기본 구성
- 관리/데이터 통합

② 네트워크 패스 이원화
- 관리/데이터 분리

③ 하드웨어 가속 기술 지원
- 관리/데이터분리&가속



6 지능형 서비스 가속 실행 (1)

- 지능형 서비스의 자원할당
  - CPU : hyper-thread 단위로 할당 (Integer 단위로 요청/할당)
  - 시스템 운영에 필요한 CPU 사용 보장 (kubelet의 config 설정)

```
kubeReserved:
  cpu: 500m
```

  - GPU-CPU 자원간 토폴로지 인지 자원 할당: Best-effort

```
Addresses:
  InternalIP:  129.254.202.130
  Hostname:    gedgeworker01
Capacity:
  cpu:                80
  ephemeral-storage:  427237720Ki
  hugepages-1Gi:      0
  hugepages-2Mi:      0
  memory:             263692960Ki
  pods:               110
Allocatable:
  cpu:                80
  ephemeral-storage:  393742282101
  hugepages-1Gi:      0
  hugepages-2Mi:      0
  memory:             263590560Ki
  pods:               110
```

```
Addresses:
  InternalIP:  129.254.202.133
  Hostname:    gedgeworker02
Capacity:
  cpu:                80
  ephemeral-storage:  427237720Ki
  hugepages-1Gi:      0
  hugepages-2Mi:      0
  memory:             263692948Ki
  pods:               110
Allocatable:
  cpu:                79500m
  ephemeral-storage:  393742282101
  hugepages-1Gi:      0
  hugepages-2Mi:      0
  memory:             263590548Ki
  pods:               110
```

서비스 정의

```
service:
  name: app2,
  containers: [ {
    image: k8s.gcr.io/echoserver:1.10,
    ports: [ {
      containerPort: 8080,
      protocol: TCP,
      externalPort: 80
    }],
    resources: {
     requests: {
      cpu: 200m,
      memory: 500Mi },
     acceleration : {
      toploygyAware: true,
      compute: [
        {"GPU" : 1}
      ]}
      communication:  SRIOV
    }
  } ]
```

7 지능형 서비스 가속 실행 (2)

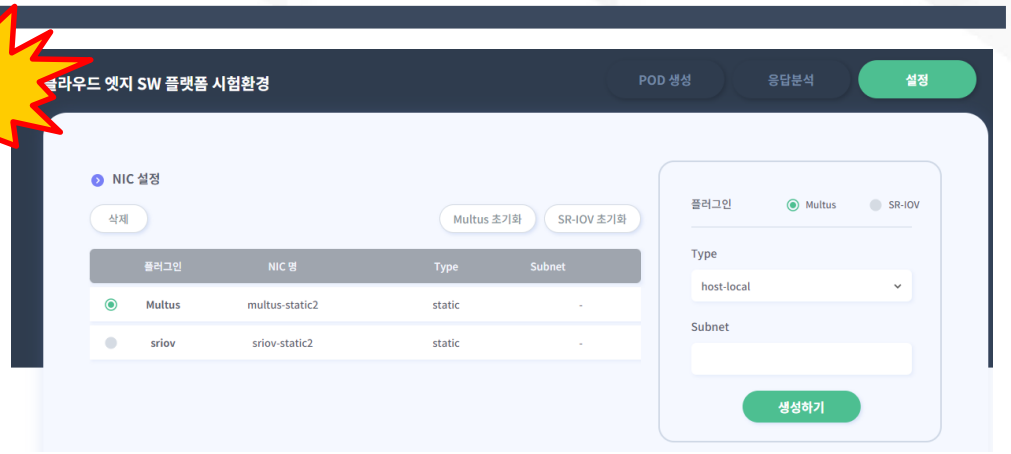
- 지능형 서비스의 데이터 전송 패스 가속
  - 이종의 데이터 전송 가속 패스 구성 지원

Network Plugin : Flannel, Multus, SRIOV, RDMA

- 네트워크 구성에 따른 성능 시험 도구 제공

SR-IOV 지원

시연



Network ConfigMap

03_sriov_configmap.yaml

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: sriovdp-config
  namespace: kube-system
data:
  config.json: |
    {
        "resourceList": [
            {
                "resourceName": "intel_sriov_netdevice",
                "selectors": {
                    "vendors": ["8086"],
                    "devices": ["37cd"],
                    "drivers": ["iavf"]
                }
            }
        ]
    }
```

Network CRD

06_sriov-static.yaml

```yaml
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: sriov-net1
  annotations:
    k8s.v1.cni.cncf.io/resourceName: intel.com/intel_sriov_netdevice
spec:
  config: '{
  "type": "sriov",
  "cniVersion": "0.3.1",
  "name": "sriov-network",
  "ipam": {
    "type": "static"
  }
}'
```

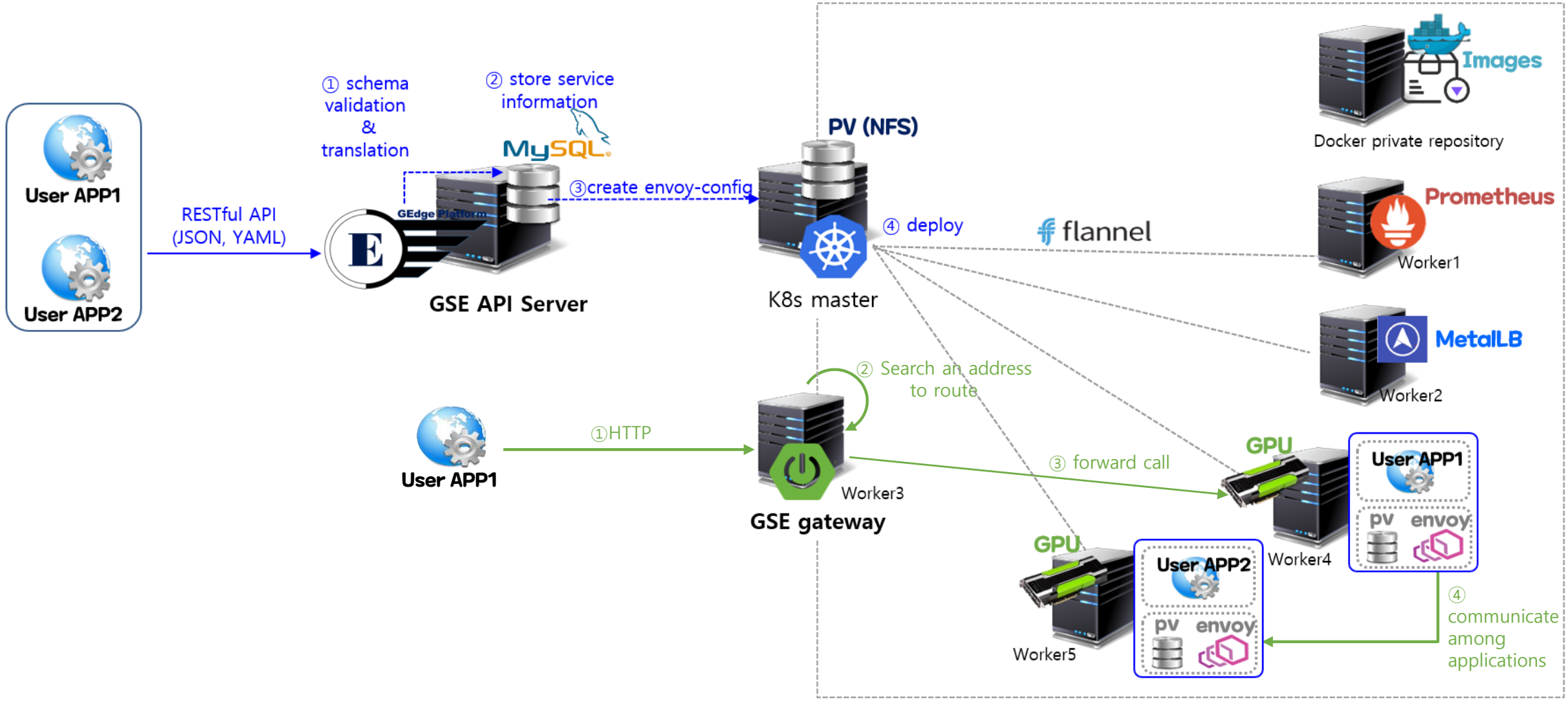
Service Definition

07_test-pods > iperf-server-sriov-1.yml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: sriov-server-1
  labels:
    name: sriov-server-1
  annotations:
    k8s.v1.cni.cncf.io/networks: '[
            { "name": "sriov-net1",
              "ips": [ "10.10.2.60/24" ]
            }
    ]'
spec:
  nodeName: gedgew01
  containers:
  - name: sriov-server-1
    image: cisdocker/iperf3:2020
    command: ["iperf3", "-s"]
    resources:
      requests:
        intel.com/intel_sriov_netdevice: '1'
      limits:
        intel.com/intel_sriov_netdevice: '1'
```

8 마이크로서비스 응용 지원 (1)

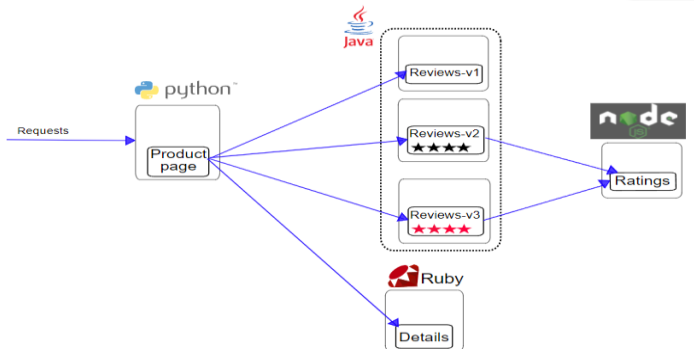
- 마이크로서비스 응용 실행 및 접근



9 마이크로서비스 응용 지원 (2)

- 마이크로서비스 응용 정의

  - Template: 여러 마이크로서비스 응용 정의에서 공통 사용 지원 (실행 공유: 미지원)
  - ServiceMesh: 다른 버전의 응용을 하나의 서비스로 정의 지원, 사용자 접근 응용 지정



Template

```
"template": {
  "name": "hw-app",
  "service": {
    "containers": [
    {
      "image": "129.254.202.122:5000/hw_app:v2.0",
      "ports": [{
        "containerPort": 8080,
        "protocol": "TCP",
        "externalPort": 80
      }],
      "resources": {
        "requests": {
          "cpu": "200m",
          "memory": "500Mi"
        }
      }
    }
    ]
} }
```

ServiceMesh

```
"serviceMesh": {
  "name": "service-mesh2",
  "services": [
   {
    "name": "product",
    "template": "bookinfo-product",
    "labels": {
     "version": "v1"
    },
    "externalAccess": true
   },
   {
    "name": "details",
    "template": "bookinfo-details",
    "labels": {
     "version": "v1"
    }
   } ],
```

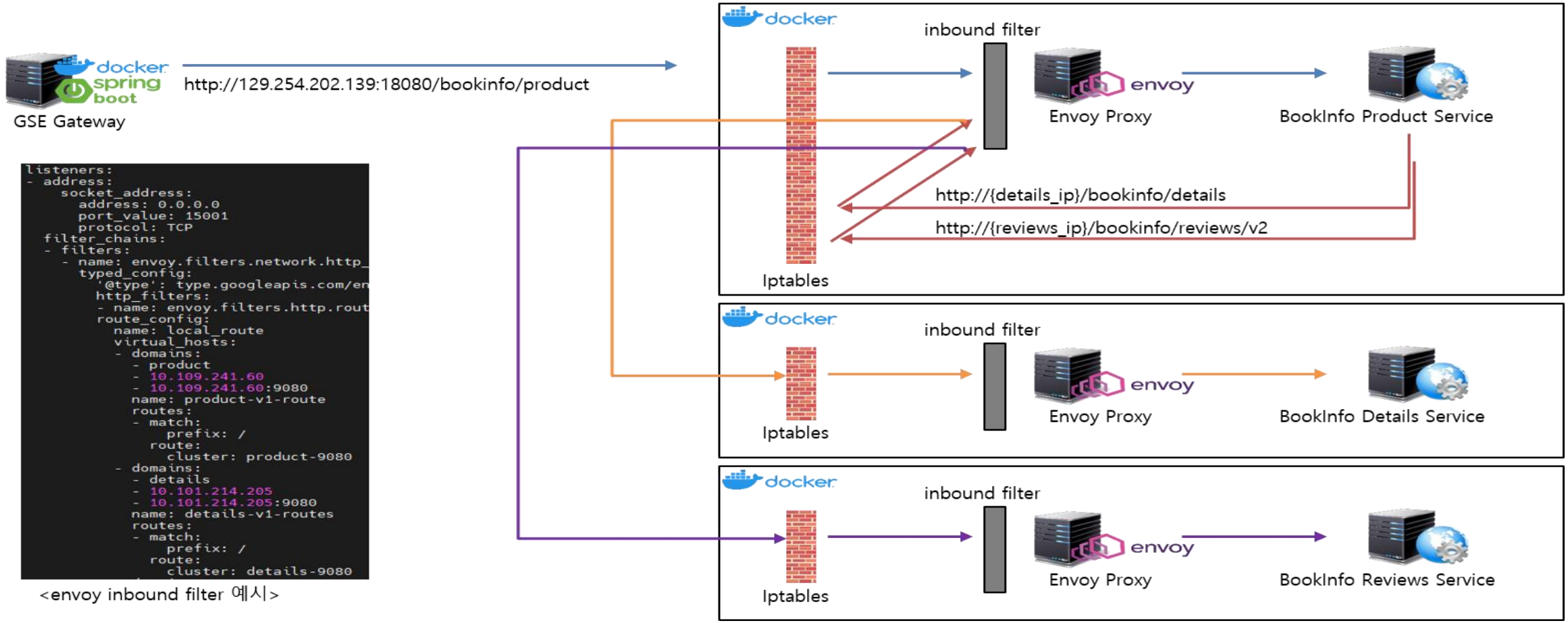
```
"serviceRoutes": [
  {
   "name": "product-v1-route",
   "match": [ {
     "uri": {
      "prefix": "/ }
   } ],
   "destination": {
     "host": "product",
     "subset": "v1"}
  }, {
   "name": "details-v1-routes",
   "match": [ {
     "uri": {
      "prefix": "/"}
   } ],
   "destination": {
     "host": "details"
   }
]}
```

시연

10 마이크로서비스 응용 지원 (3)

- 마이크로서비스 응용 연계

  - Init container를 이용한 iptables 설정




```
listeners:
- address:
    socket_address:
      address: 0.0.0.0
      port_value: 15001
      protocol: TCP
  filter_chains:
  - filters:
    - name: envoy.filters.network.http_
      typed_config:
        '@type': type.googleapis.com/en
        http_filters:
        - name: envoy.filters.http.rout
        route_config:
          name: local_route
          virtual_hosts:
          - domains:
            - product
            - 10.109.241.60
            - 10.109.241.60:9080
            name: product-v1-route
            routes:
            - match:
                prefix: /
              route:
                cluster: product-9080
          - domains:
            - details
            - 10.101.214.205
            - 10.101.214.205:9080
            name: details-v1-routes
            routes:
            - match:
                prefix: /
              route:
                cluster: details-9080
```

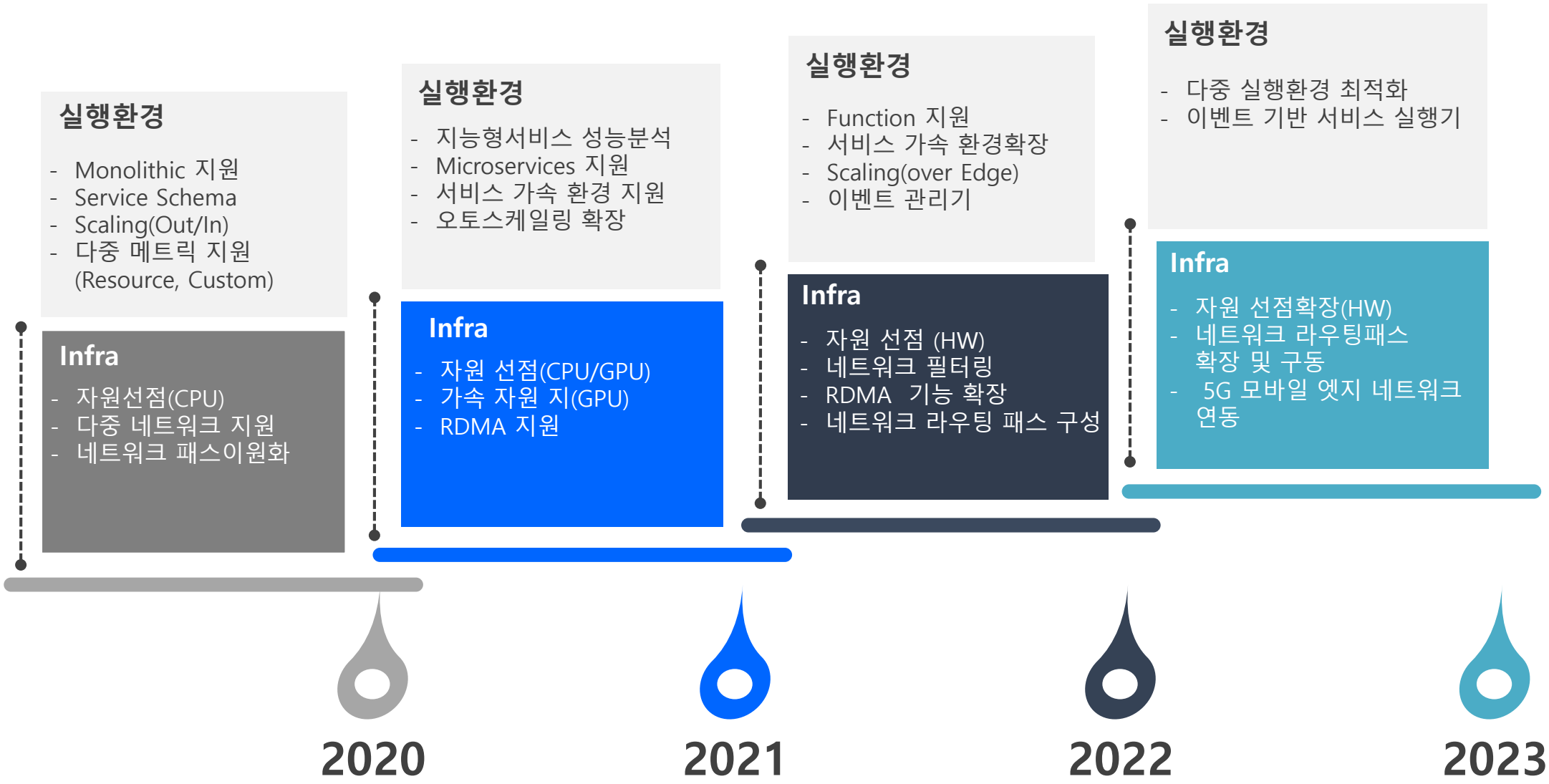
<envoy inbound filter 예시>

III 기술 데모 및 향후 계획

1 기술 데모


2종 실행환경 지원
모놀리딕/마이크로서비스 응용 실행
실행환경
엣지 서비스 성능
네트워크 성능 시험 도구
네트워크 가속

2 향후 계획


실행환경
- Monolithic 지원
- Service Schema
- Scaling(Out/In)
- 다중 메트릭 지원 (Resource, Custom)
Infra
- 자원선점(CPU)
- 다중 네트워크 지원
- 네트워크 패스이원화
2020
실행환경
- 지능형서비스 성능분석
- Microservices 지원
- 서비스 가속 환경 지원
- 오토스케일링 확장
Infra
- 자원 선점(CPU/GPU)
- 가속 자원 지(GPU)
- RDMA 지원
2021
실행환경
- Function 지원
- 서비스 가속 환경확장
- Scaling(over Edge)
- 이벤트 관리기
Infra
- 자원 선점 (HW)
- 네트워크 필터링
- RDMA 기능 확장
- 네트워크 라우팅 패스 구성
2022
실행환경
- 다중 실행환경 최적화
- 이벤트 기반 서비스 실행기
Infra
- 자원 선점확장(HW)
- 네트워크 라우팅패스 확장 및 구동
- 5G 모바일 엣지 네트워크 연동
2023

감사합니다.

http://gedge-platform.github.io

GS-Engine 프레임워크 리더
최현화(hyunwha@etri.re.kr)