



GEdge Platform

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

지능형 엣지 서비스 실행가속을 위한 SW/인프라 기술

2022.12.20

GS-Engine 프레임워크 코어개발자

차재근

(jocha@etri.re.kr)
“GEdge Platform”은 클라우드 중심의 엣지 컴퓨팅 플랫폼을 제공하기 위한
핵심 SW 기술 개발 커뮤니티 및 개발 결과물의 코
드명입니다.
Developer-Driven

GEdge Platform Community 5th Conference (GEdge Platform v3.0 Release) -

Contents

- I** GS-Engine 개요
- II** 데이터 처리/전송 가속 기술
- III** 실행 환경 최적화 기술
- IV** GS-Engine 기술 적용 서비스

GEdge 플랫폼 내 GS-Engine의 포지셔닝

초저지연 지능형 클라우드 엣지 플랫폼 (GEdge Platform)

클라우드 엣지 관리 플랫폼 (GM : GEdge Management Platform)

플랫폼 관리 도구 프레임워크 (GM-Tool)

Framework I/F

플랫폼 관리 기능 프레임워크 (GM-Center)

Platform I/F

지능형 서비스 운용 프레임워크 (GS-AI)

엣지 AI 서비스 환경
(GS-AIflow)

엣지 협업 학습 환경
(GS-Optops)

서비스 협업 프레임워크 (GS-Link)

협업 게이트웨이
(GS-Linkgw)

협업 정책 생성
(GS-Linkhq)

Framework I/F

Framework I/F

초저지연 데이터 처리 프레임워크 (GS-Engine)

엣지 전용 스케줄러
(GS-Scheduler)

엣지 메시지 브로커
(GS-Broker)

클라우드 엣지 서비스 플랫폼 (GS : GEdge Service Platform)

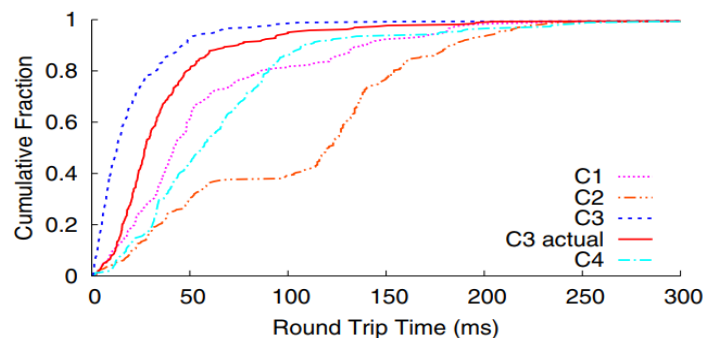


I

GS-Engine 개요



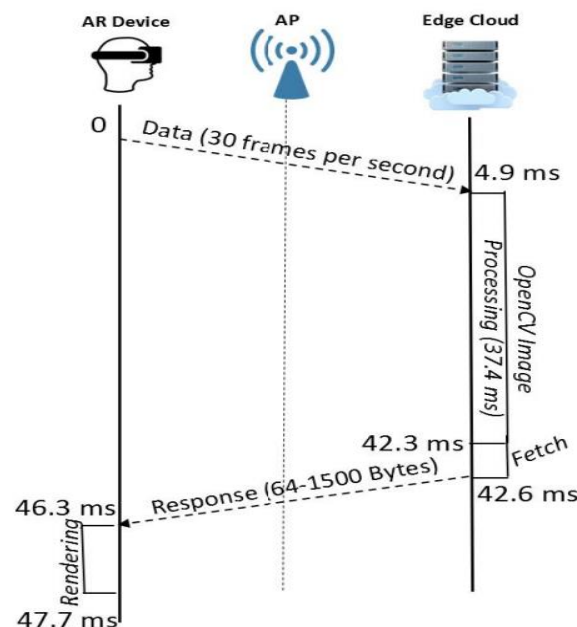
Public Cloud



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

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

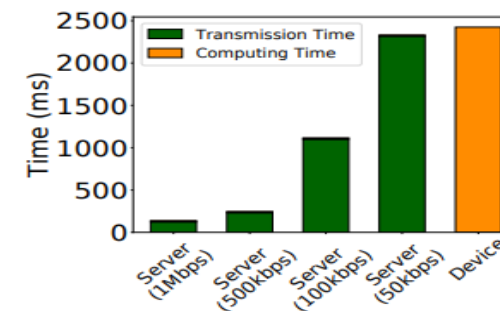
Edge Cloud



The latency of services has **50ms**

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

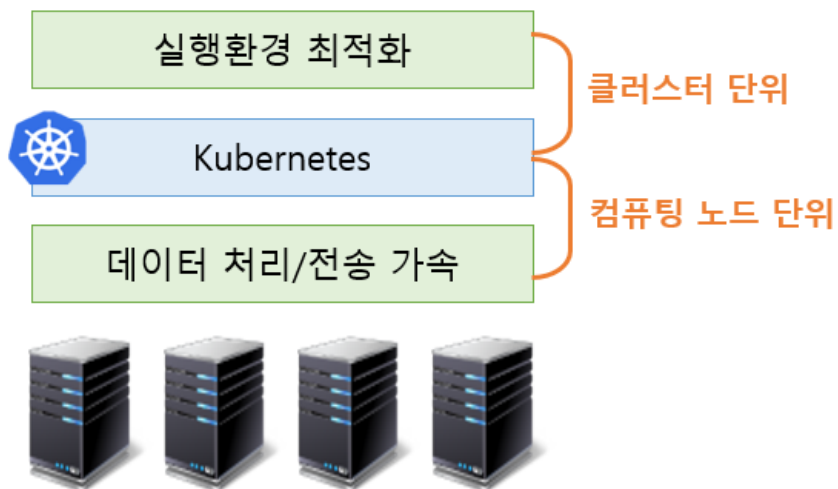
Smart Device



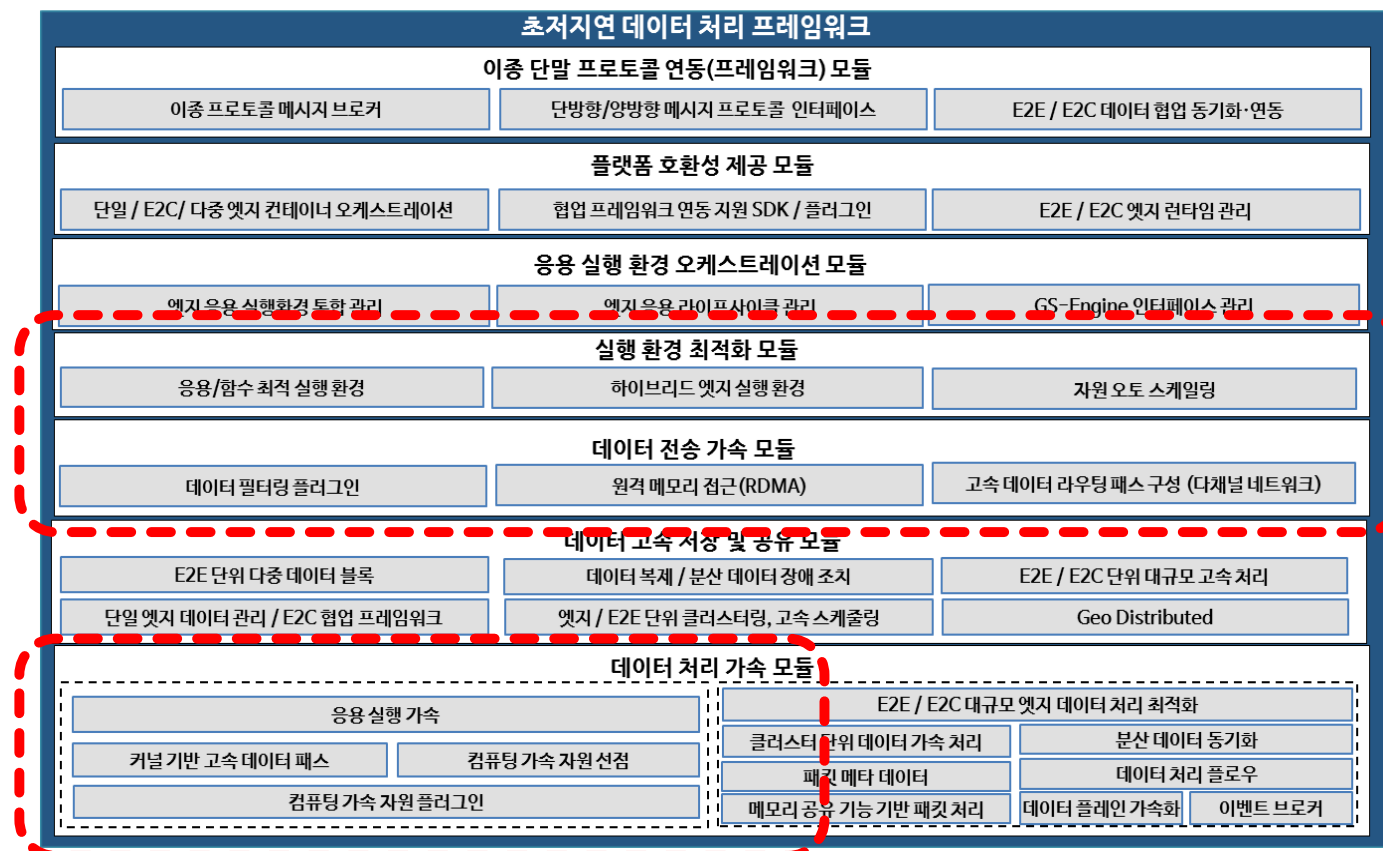
It takes **more than 2s** to execute the model on the resource-limited Raspberry Pi

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

- 대규모 데이터 사용 엣지 서비스 지원 ← SW 인프라/가속 기술 바탕의 실행환경 관리
 - 데이터 처리 가속 모듈 : 서비스별 컴퓨팅 자원 할당, 시스템 SW 기반 컴퓨팅 자원선점 → 서비스 처리 성능 가속
 - 데이터 전송 가속 모듈 : 입출력 데이터 필터링, 데이터 전송 패스 이원화 → 저지연 데이터 전송
 - 실행환경 최적화 모듈 : 모놀리식, 마이크로서비스 등 이종 응용 실행 지원 → 다양한 엣지 서비스 운용 지원



<GS-Engine 실행 단위 및 구성도>

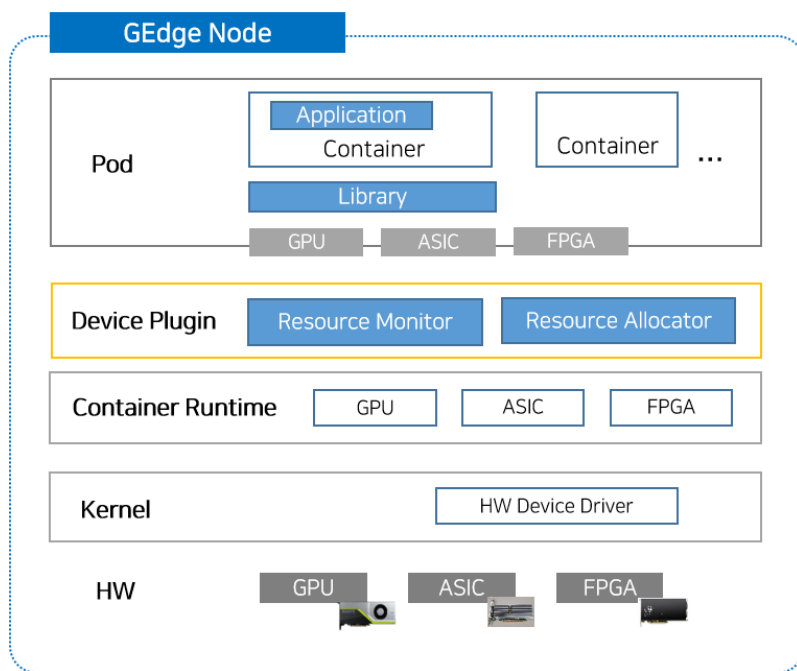


I

데이터 처리/전송 가속 기술



- 컨테이너(포드) 환경에 하드웨어 가속 장치 할당 및 활용
 - GPU(Nvidia), ASIC, FPGA(Intel Stratix) 장치 연동
- 장치 플러그인 형태 구성
 - 자원 할당 및 모니터링
 - 장치 설정 및 할당 절차 간소화



〈GS-Engine 컴퓨팅 가속 자원 플러그인 개념도〉

① 컨테이너 관리 플랫폼 내 GPU 할당

```
root@gedgemaster:~# kubectl get nodes "-o=custom-columns=NAME:.metadata.name,GPU:.status.allocatable.nvidia\.com/gpu"
```

NAME	GPU
gedgemaster	<none>
gedgeworker1	1
gedgeworker2	1
gedgeworker3	1

② GPU 활용 컨테이너(포드) 및 서비스 생성

```
root@cnode19:~/k8s-deeplearning-example# kubectl get svc,rc,pods
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/keras-app-nodeport	NodePort	10.96.70.84	<none>	80:31614/TCP	19h
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	7d15h

NAME	DESIRED	CURRENT	READY	AGE
replicationcontroller/keras-rc	1	1	1	19h

NAME	READY	STATUS	RESTARTS	AGE
pod/keras-rc-czdc2	1/1	Running	1	19h

③ 이미지 분석 실행 결과

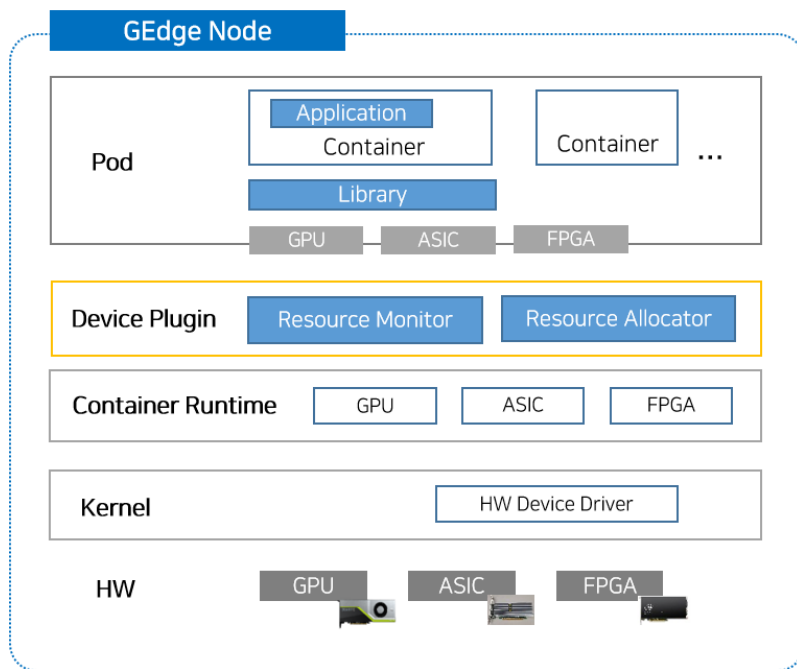


```
[root@imvdiserver ~]# curl -X POST -F image=@cat.jpg 'http://129.254.202.19:31614/predict'
```

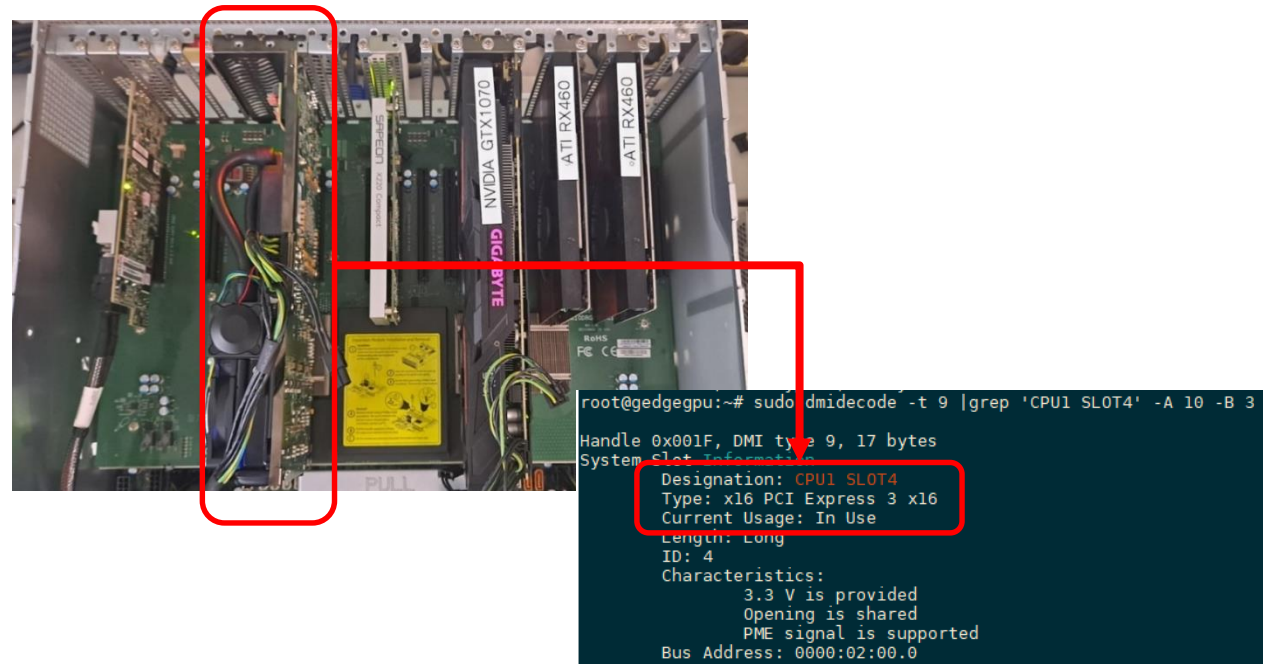
```
{
  "predictions": [
    { "label": "tiger_cat", "probability": 0.8349542021751404 },
    { "label": "tabby", "probability": 0.0915290042757988 },
    { "label": "Egyptian_cat", "probability": 0.0306521188467741 },
    { "label": "quilt", "probability": 0.013612422160804272 },
    { "label": "lynx", "probability": 0.005530184600502253 }
  ],
  "success": true
}
```

〈GS-Engine 컴퓨팅 자원 할당 및 서비스 활용 예 - GPU〉

- 컨테이너(포드) 환경에 하드웨어 가속 장치 할당 및 활용
 - GPU(Nvidia), ASIC, FPGA(Intel Stratix) 장치 연동
- 장치 플러그인 형태 구성
 - 자원 할당 및 모니터링
 - 장치 설정 및 할당 절차 간소화

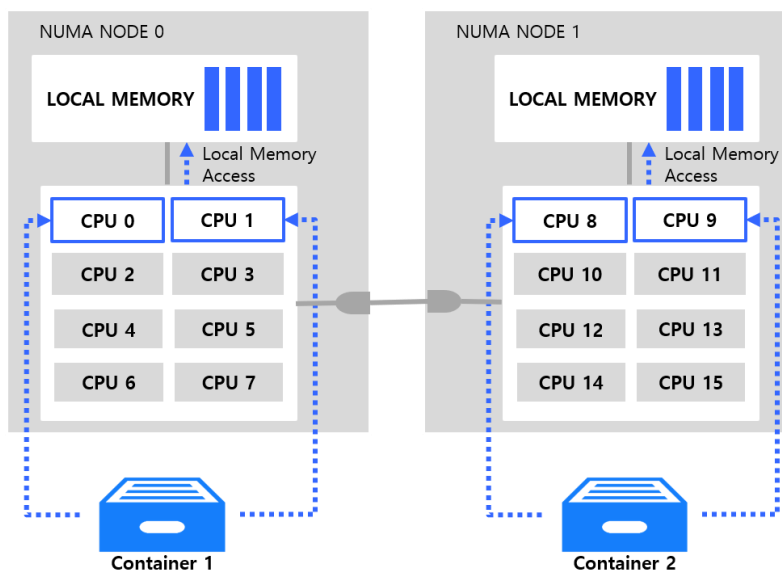


〈GS-Engine 컴퓨팅 가속 자원 플러그인 개념도〉



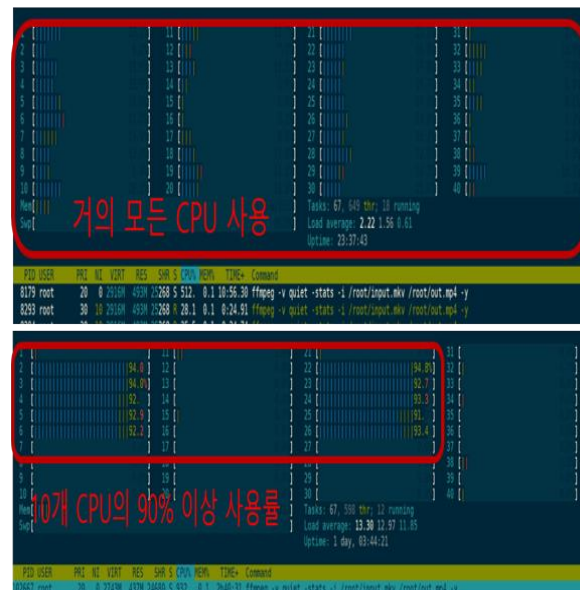
〈GS-Engine 컴퓨팅 자원 구성 형상 - FPGA〉

- CPU 자원 선점 (CPU pinning)
 - 지정된 CPU 코어를 서비스에 지정 할당하여 Context Switching ↓, 데이터 처리 성능 ↑
- CPU Intensive Application 경우,
 - 동영상 인코딩 : 52% 성능 향상 (2,252초 → 1,290초)
 - 객체 인식: 300% 성능 향상 (20.58분 → 7.18분)

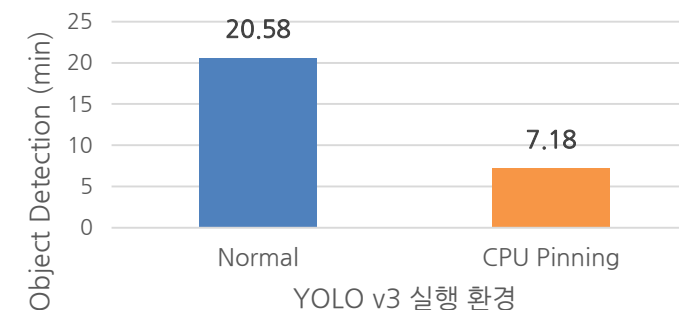


〈CPU 자원 선점 개념도〉

엣지 서비스의 CPU 선점



CPU 선점 기반 Video 분석

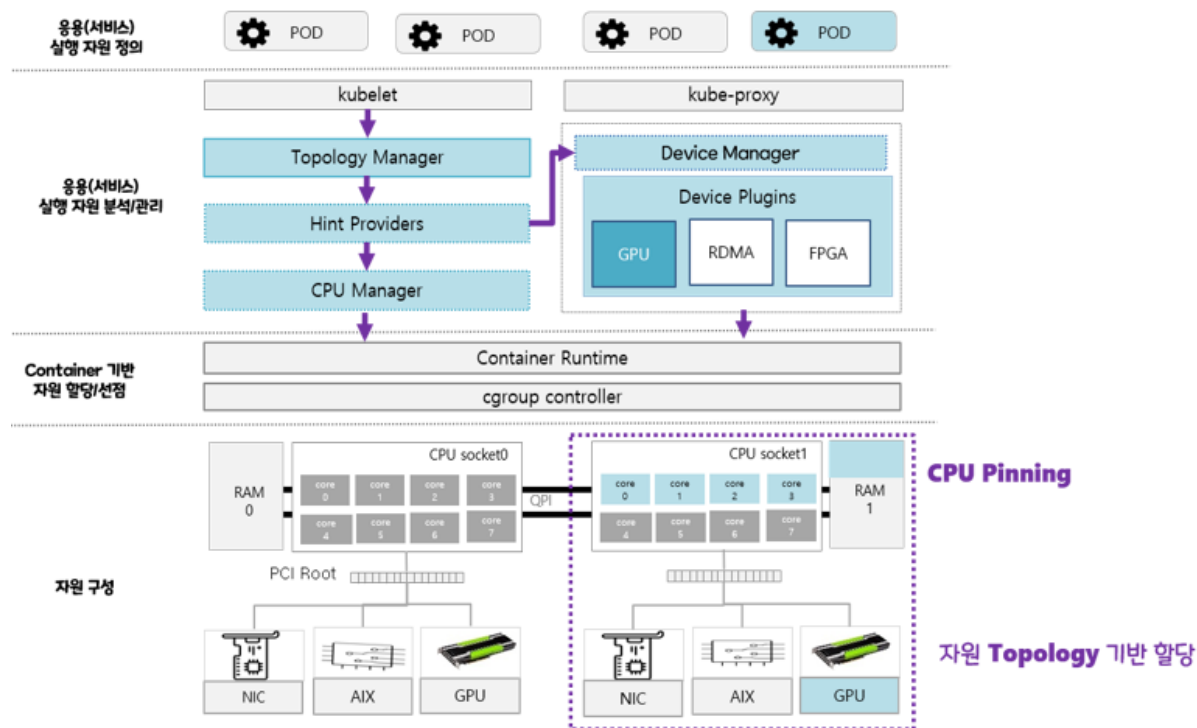


YOLO v3 기반 Scooter.mp4 분석
 → HD(1280*720), 14sec(30.481KB)
 Sliding Window 분석 속도 (1 FPS)
 → 약 300% 성능 향상



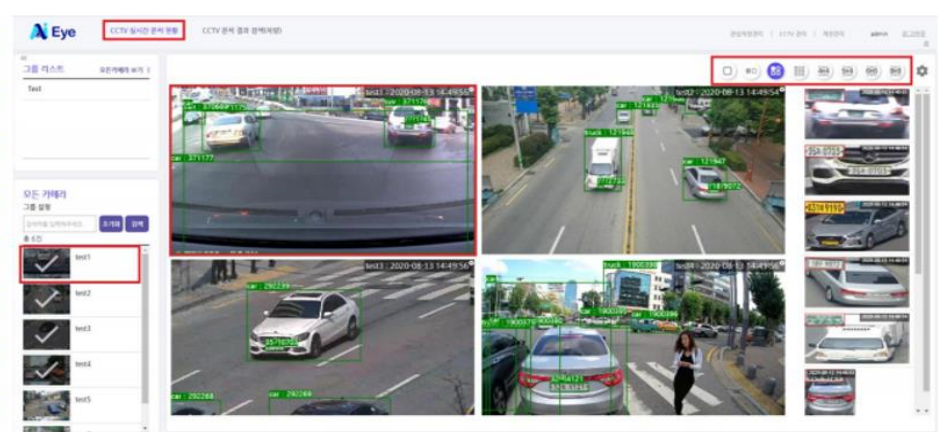
〈CPU 자원선점(Pinning) 테스트 결과〉

- 토폴로지(Topology) 인지형 자원 선점
 - 대규모 지능형 서비스는 CPU/GPU/NIC 등 여러 장치 협업, 여러 개의 AI 모델 사용
 - ex) CCTV와 같은 스트리밍 분석
 - 이 때, 컴퓨팅 자원의 Affinity 확보, Data Copy ↓



〈GS-Engine 토폴로지 인지형 컴퓨팅 자원 선점 기술〉

Vehicle Plate Analysis



SW requirement

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

〈자원 선점 기술 성능 시험에 활용한 자동차 분석 서비스〉

- 자원 선점 성능 분석 서비스 개요
 - Data: FHD(1920*1080), 709 MB, 107,836 Frames
 - Model: SSD_MobileNet v2, LeNet-PReLU, DeepSORT
- I/O Intensive Application 경우
 - CPU 자원 선점 시, 엣지 서비스 성능 향상(3%)
 - 대용량 데이터 입력, 다중 모델(데이터 이동)/이종 자원 → PCIe 공유자원 Bottleneck 발생

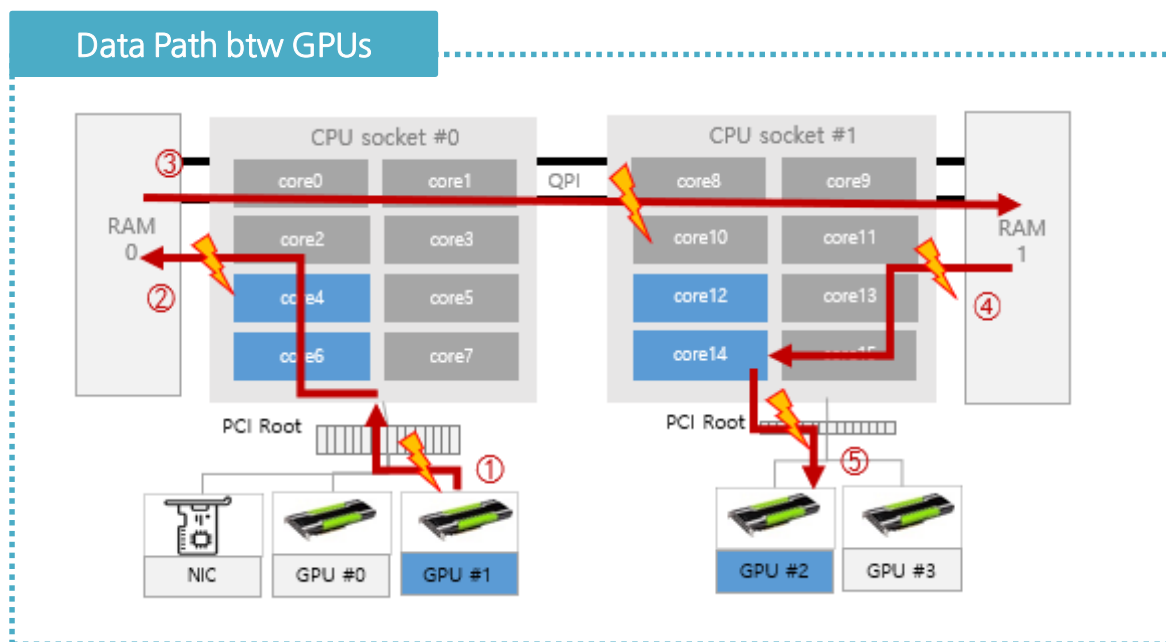
① 성능 실험 결과 (1 container, 1 노드)

	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

② 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

- 데이터 처리 성능 향상 요소: 이종 자원 간의 데이터 이동/복사 감소
- Resource Distance Matrix
 - Data 전송 패스(Path)를 바탕으로 자원 할당
 - 다중 지능형 서비스의 동시 실행을 위한 전송 패스의 이원화
 - 마이크로서비스 실행을 위한 자원 할당 알고리즘 활용

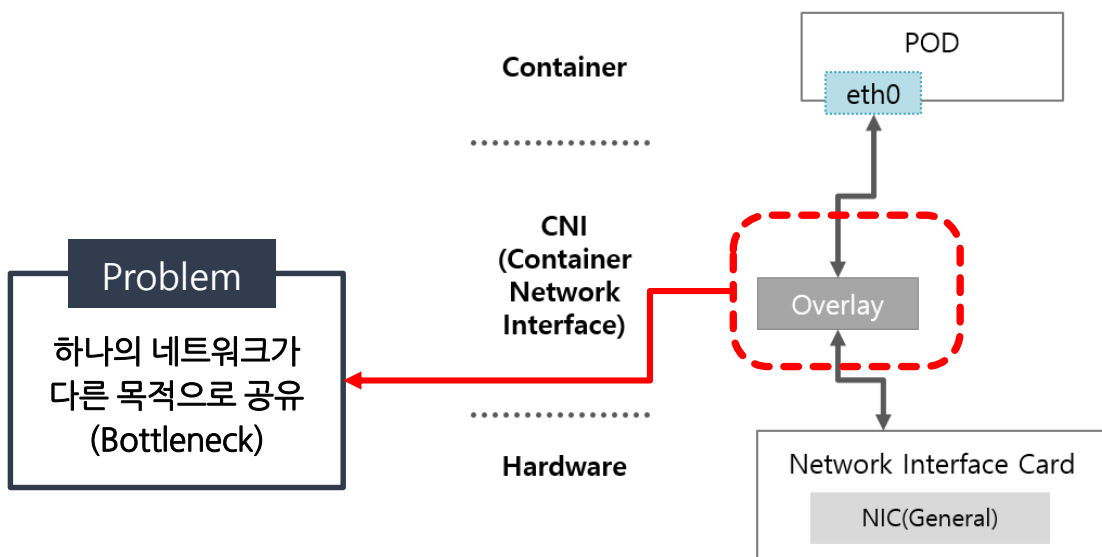


<Resource Distance Matrix의 분석 지점>

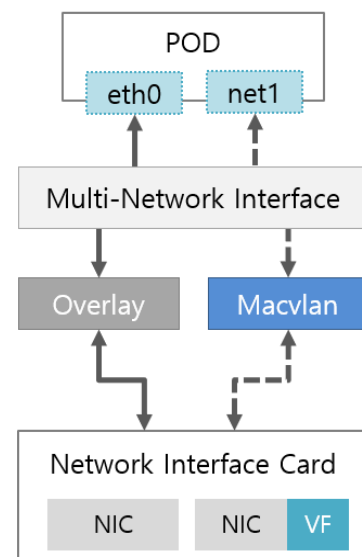
	NIC	GPU0 (Dist)	GPU1 (Dist)	GPU2 (Dist)	SSD (Dist)	GPU3 (Dist)
NIC	X	PIX (2)	NODE (4)	NODE (4)	SYS (5)	SYS (5)
GPU0	PIX (2)	X	PXB (3)	PXB (3)	SYS (5)	SYS (5)
GPU1	PXB (3)	PXB (3)	X	PIX (2)	SYS (5)	SYS (5)
GPU2	PXB (3)	PXB (3)	PIX (2)	X	SYS (5)	SYS (5)
SSD	SYS (5)	SYS (5)	SYS (5)	SYS (5)	X	PIX (2)
GPU3	SYS (5)	SYS (5)	SYS (5)	SYS (5)	PIX (2)	X
GPU4	SYS (5)	SYS (5)	SYS (5)	SYS (5)	PXB (3)	PXB (3)
GPU4	SYS (5)	SYS (5)	SYS (5)	SYS (5)	PXB (3)	PXB (3)

<Resource Distance Matrix 분석 결과>

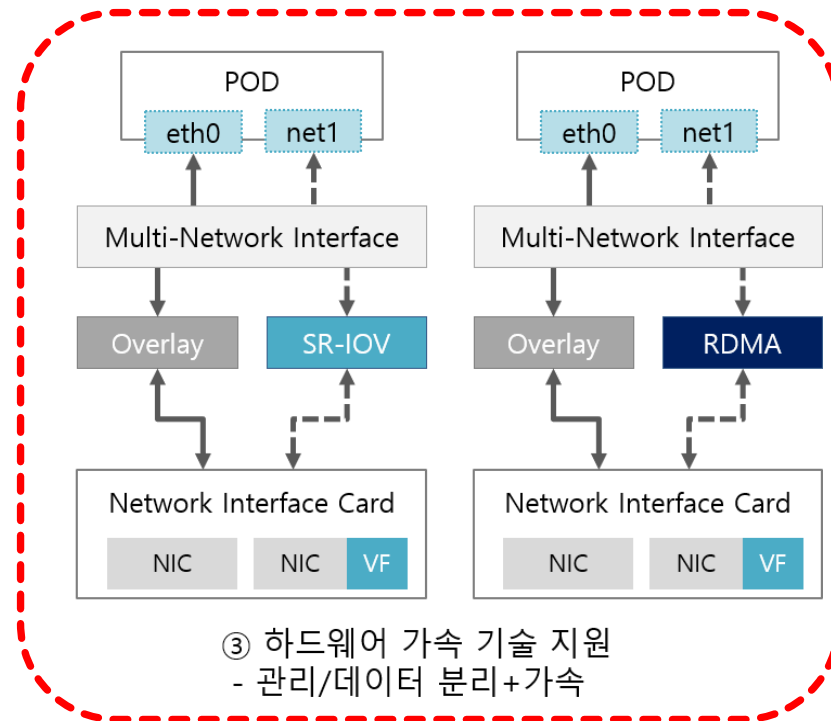
- 네트워크 장치 하드웨어 가상화/가속 기술 활용 데이터 전송 경로 이원화
 - 컨테이너 내부 가상 네트워크 장치 추가 → 데이터 전송 패스 이원화
 - 컨테이너에 가상 네트워크 장치 추가 시, 네트워크 장치 가속 기능 제공(SR-IOV)
 - 컨테이너 내 애플리케이션의 고속 데이터 접근 기능, RDMA 활용지원



① 기본 구성
- 관리/데이터 통합



② 네트워크 패스 이원화
- 관리/실행 데이터 분리



③ 하드웨어 가속 기술 지원
- 관리/데이터 분리+가속

<데이터 전송 가속 기술(②,③) 개념도>

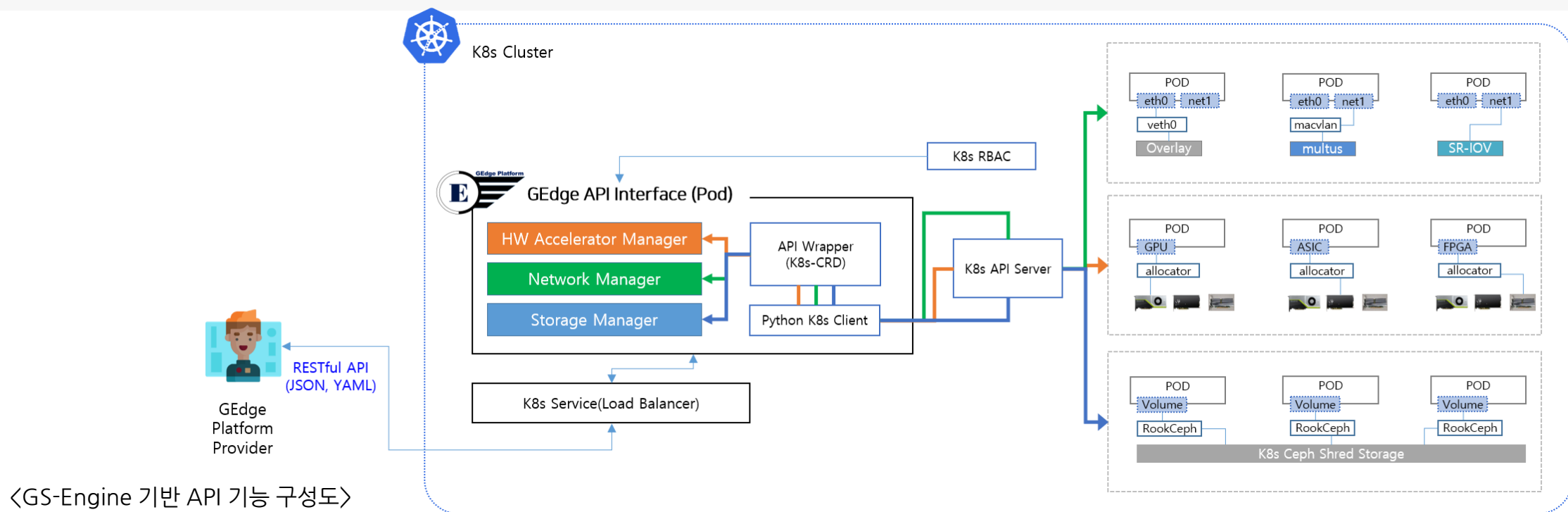


III

실행 환경 최적화 기술



- 서비스 실행 및 가속 자원 구성 편의 제공을 위한 인터페이스 제공(RESTful API)
 - 하드웨어 가속 자원(GPU, ASIC, FPGA) 구성 및 관리 기능
 - 컴퓨팅 자원 선점(토폴로지 매니저)
 - 데이터 전송 패스 이원화(다중 NIC, SR-IOV) 자원 구성 및 관리 기능
 - 마이크로서비스 기반 서비스 메시 구성 기능
 - 공용 데이터 처리용 공유 스토리지 자원 구성 및 관리 기능



- 데이터 처리/전송 가속 관련 서비스 스키마 구성 → Acceleration
- 지능형 서비스의 자원 할당 방법
 - CPU: Hyper-thread 단위 할당 (Integer 단위 요청, 할당)
 - CPU-GPU 자원 간 Topology 인지는 Best-effort 로 수행

```
spec:
  containers:
  ...
  - name: app
    resources:
      requests:
        memory: "50Mi"
        cpu: "250m"
      limits:
        memory: "100Mi"
        cpu: "500m"
    Acceleration:
      topologyAware: true
      communication: '["SRIOV", "RDMA"]'
      compute: '["GPU, FPGA, AIX..."]'
    Volumes:
    - name: gedgevolum
```

① 자원 선점 X

```
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
```

② 자원 선점 O

```
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
```

- 컨테이너 관리 플랫폼의 서비스 스키마 단순화 → 엣지 서비스를 위한 기능 추출/추가/최적화
- 마이크로 서비스 스키마 지원
 - Service Template : 여러 마이크로 서비스 응용 정의서에서 공통으로 사용하는 부분
 - Service Mesh: 단일 목적 서비스를 제공하는 마이크로 서비스의 K8s-Service 들의 집합
 - Service Mesh는 K8s-Service 들 중, 외부에 노출 시키는 Service와 내부 호출용 Service 구분

```

"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"
          }
        }
      }
    ]
  }
}

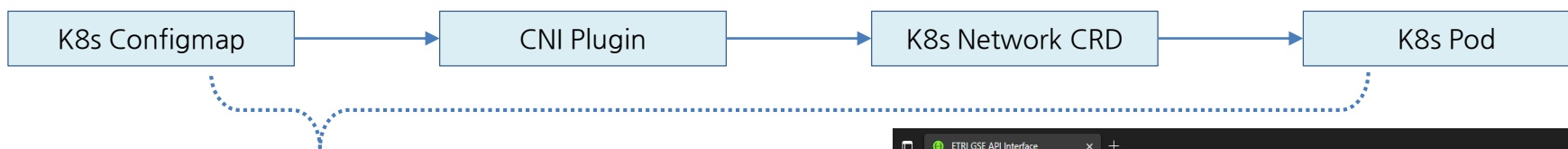
```

```

"serviceMesh": {
  "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"
      }
    }
  ]
}

```

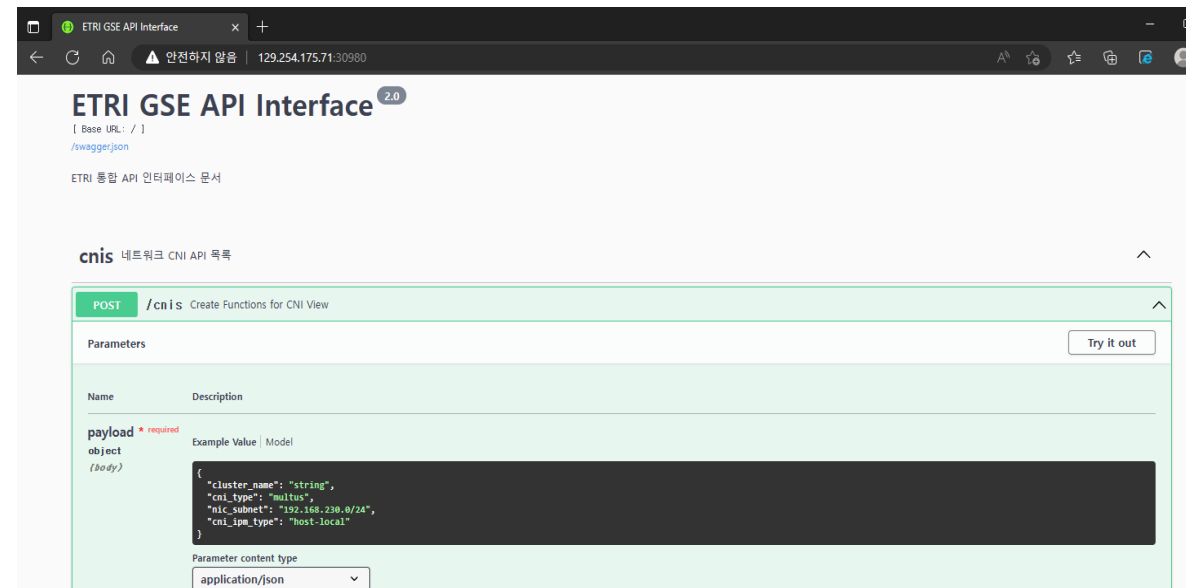
- 데이터 처리/전송 가속 기능 시험 도구 제공
 - 이종의 네트워크 전송 가속 패스 구성 지원
 - 모놀리식/마이크로 서비스 배포 및 트래픽 확인 기능 제공
 - 네트워크 설정/자원 생성/서비스 정의 단순화



```

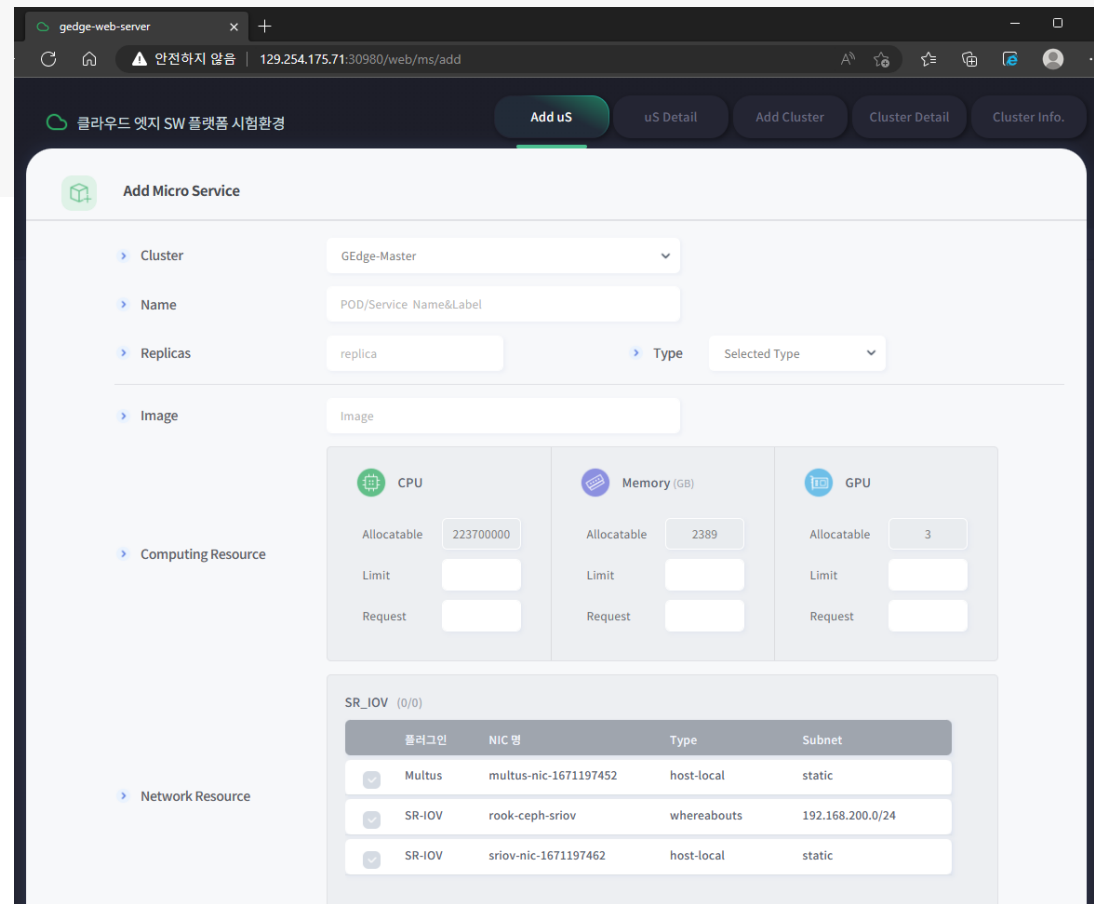
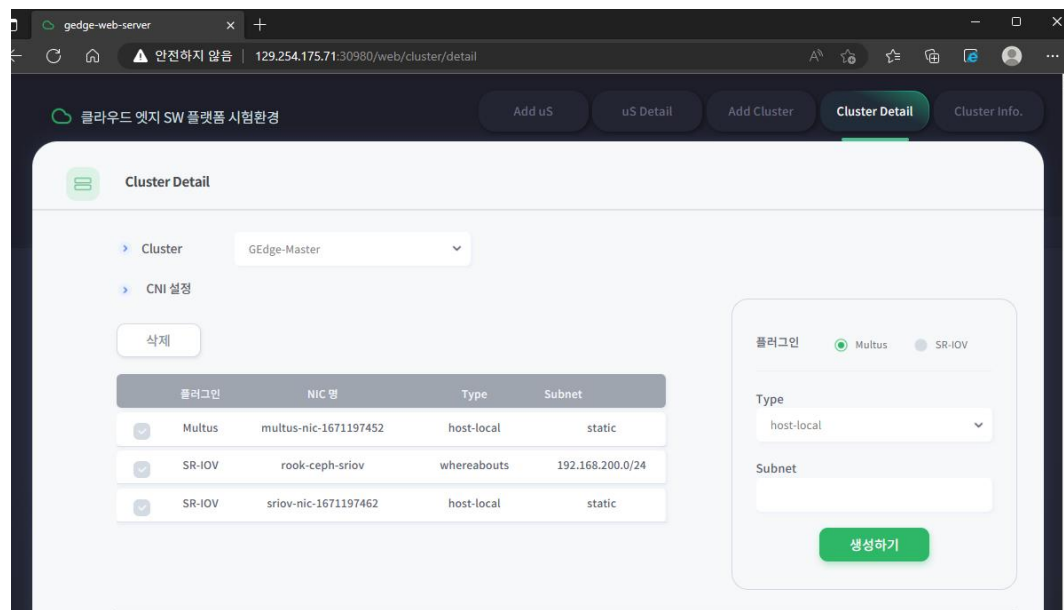
03_sriov_configmap.yaml
1  apiVersion: v1
2  kind: ConfigMap
3  metadata:
4    name: sriovdp-config
5    namespace: kube-system
6  data:
7    config.json: |
8      {
9        "resourceList": [
10         {
11           "resourceName": "intel_sriov_netdevice",
12           "selectors": {
13             "vendors": ["8086"]
14           }
15         }
16       ]
17     }
18
19
06_sriov-static.yaml
1  apiVersion: "k8s.cni.cncf.io/v1"
2  kind: NetworkAttachmentDefinition
3  metadata:
4    name: sriov-net1
5    annotations:
6      k8s.v1.cni.cncf.io/resourceName: intel.com/intel_sriov_netdevice
7  spec:
8    config: '{
9      "type": "sriov",
10     "cniVersion": "0.3.1",
11     "name": "sriov-network",
12     "ipam": {
13       "type": "static"
14     }
15   }'
16
07_test-pods > ! iperf-server-sriov-1.yaml
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: sriov-server-1
5  labels:
6    name: sriov-server-1
7  annotations:
8    k8s.v1.cni.cncf.io/networks: '[
9     {
10       "name": "sriov-net1",
11       "ips": [ "10.10.2.60/24" ]
12     }
13   ]'
14  spec:
15    nodeName: gedgew01
16    containers:
17     - name: sriov-server-1
18       image: cisdocke/iperf3:2020
19       command: ["iperf3", "-s"]
20       resources:
21         requests:
22           intel.com/intel_sriov_netdevice: '1'
23         limits:
24           intel.com/intel_sriov_netdevice: '1'
  
```

<데이터 전송 가속 네트워크 설정 예>



<데이터 전송 가속 네트워크 설정 단순화>

- 데이터 처리/전송 가속 기능 시험 도구 제공
 - 이종의 네트워크 전송 가속 패스 구성 지원
 - 모놀리식/마이크로 서비스 배포 및 트래픽 확인 기능 제공
 - 네트워크 설정/자원 생성/서비스 정의 단순화



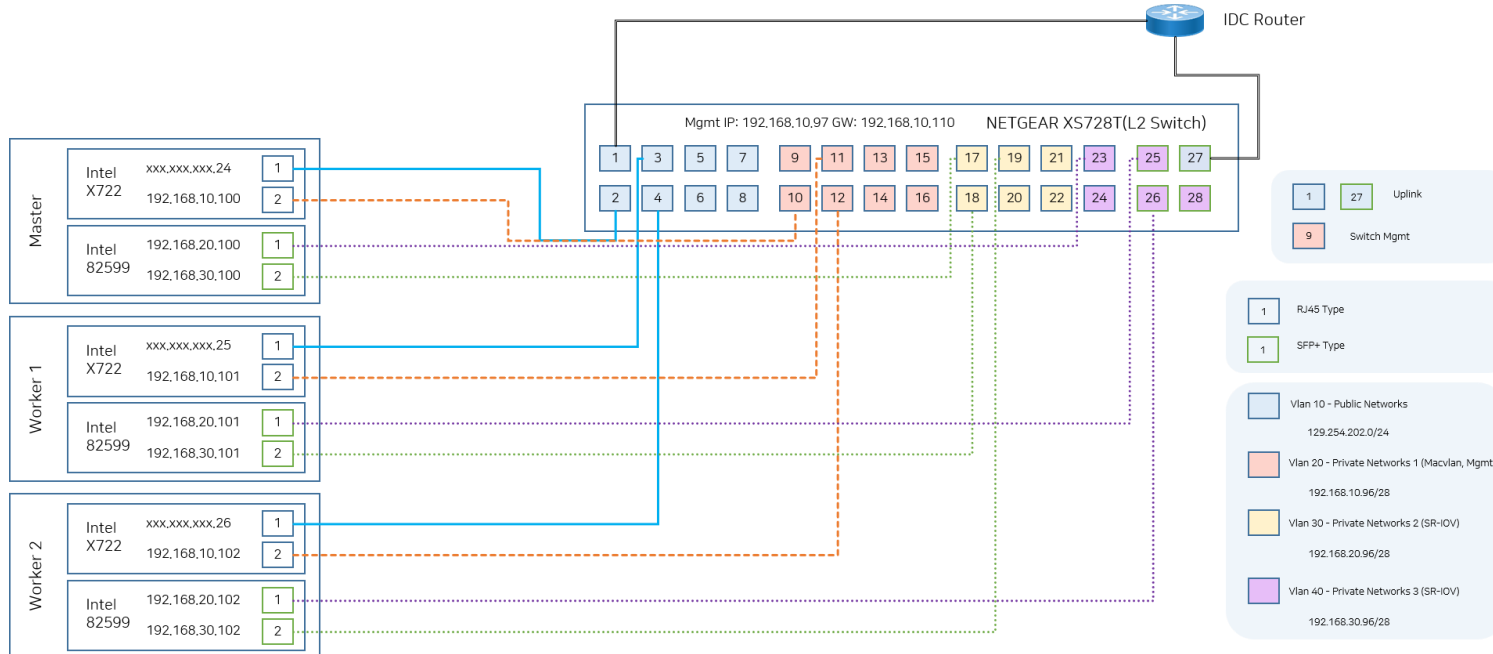
IV GS-Engine 기술 적용 서비스



- 지역별 테스트베드 구성
 - 상용 네트워크 망 : 서울(동작), 고양(일산)
 - 5G 오픈테스트랩 : 성남(판교), 대전
- 데이터 전송 가속을 위한 데이터 이원화 지원 네트워크 망 구성
 - 네트워크 장치(2ea), 개별 장치 별 포트(2ea) 활용 VLAN 구분
 - 개별 노드별 네트워크 장치 가상화 및 가속 기능 설정, 서비스에 활용



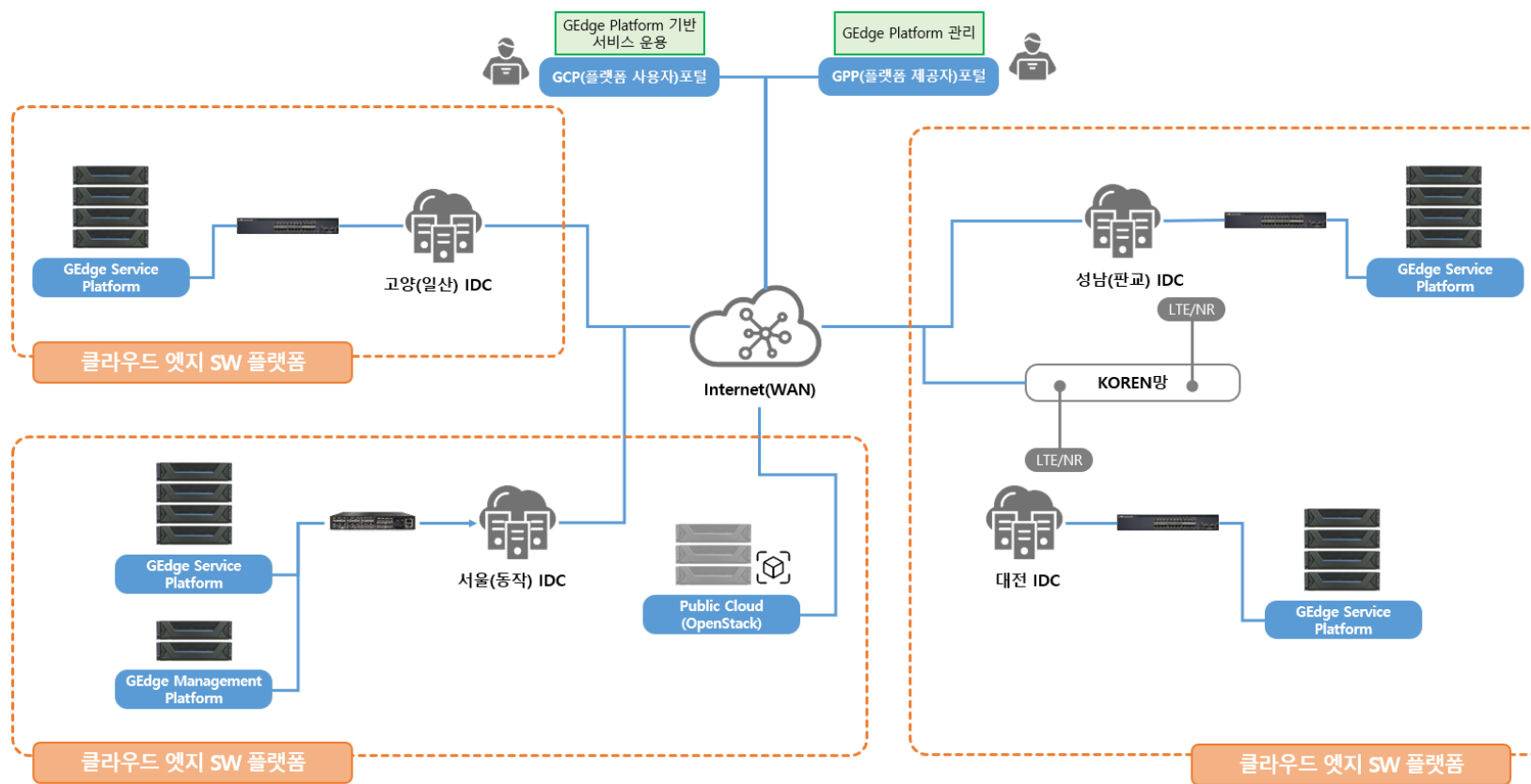
L2 Switch

 쿠버네티스 클러스터
 - Master
 - Worker 1
 - Worker 2
 - TBD


<GEdge 테스트베드 구성 모습>

<다채널 네트워크 기반 테스트베드 구축 형상>

- 지역별 테스트베드 구성
 - 상용 네트워크 망 : 서울(동작), 고양(일산)
 - 5G 오픈테스트랩 : 성남(판교), 대전



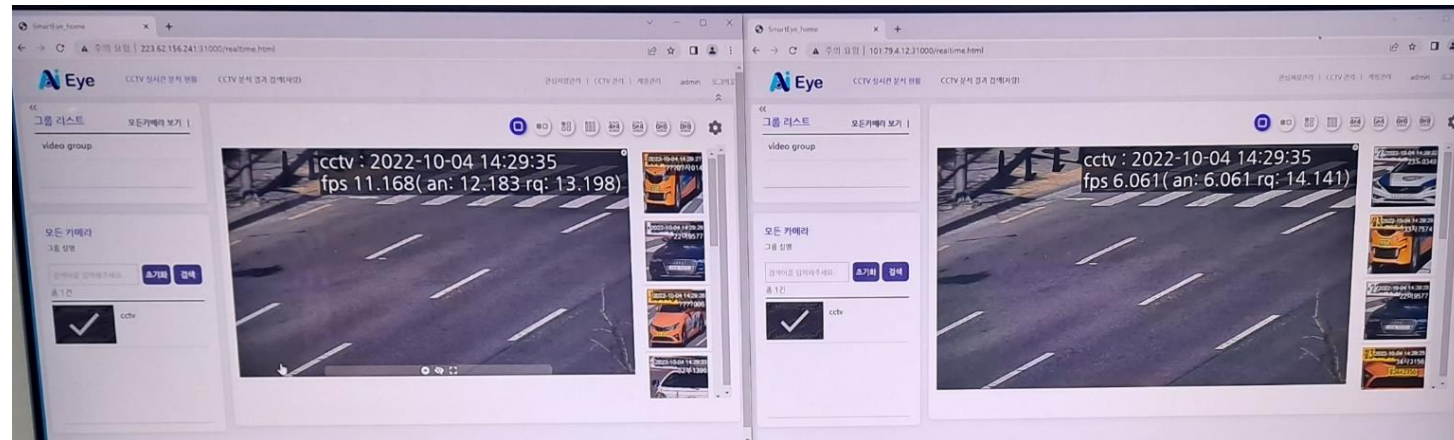
- GEdge 구축 테스트베드 활용 서비스 형상 전시
 - 22.04: ICT 사업화전시회, 22.09: 4차 산업혁명 페스티벌, 22.10: 한국 전자전(KES2022)
- 지능형 엣지 서비스 실행가속을 위한 SW/인프라 기술 적용 차량 번호판 인식 서비스 운용
 - 토폴로지 인지형 자원선점, 네트워크 데이터 패스 이원화 및 장치 가속 기능 적용
 - 분석 프레임 성능 기준의 약 2~3배 향상 확인



<GEdge 플랫폼 기반 서비스 구성 형상(전시회)>

① 실행가속 지원 분석 프레임 수 : 11.168 fps

② 기본 분석 프레임 수 : 6.061 fps



<실행가속 기술 적용 지능형 서비스의 분석 프레임 수 비교>

감사합니다.

<http://gedge-platform.github.io>



Gs-Engine 프레임워크 코어 개발자
차재근(jgcha@etri.re.kr)

Welcome to GEdge Platform

An Open Cloud Edge SW Platform to enable Intelligent Edge Service

GEdge Platform will lead Cloud-Edge Collaboration