# 엣지 서비스 실행 가속을 위한 SW/인프라기술(GS-Engine)

2020.12.10

GS-Engine 코어 개발자
최현화(hyunwha@etri.re.kr)

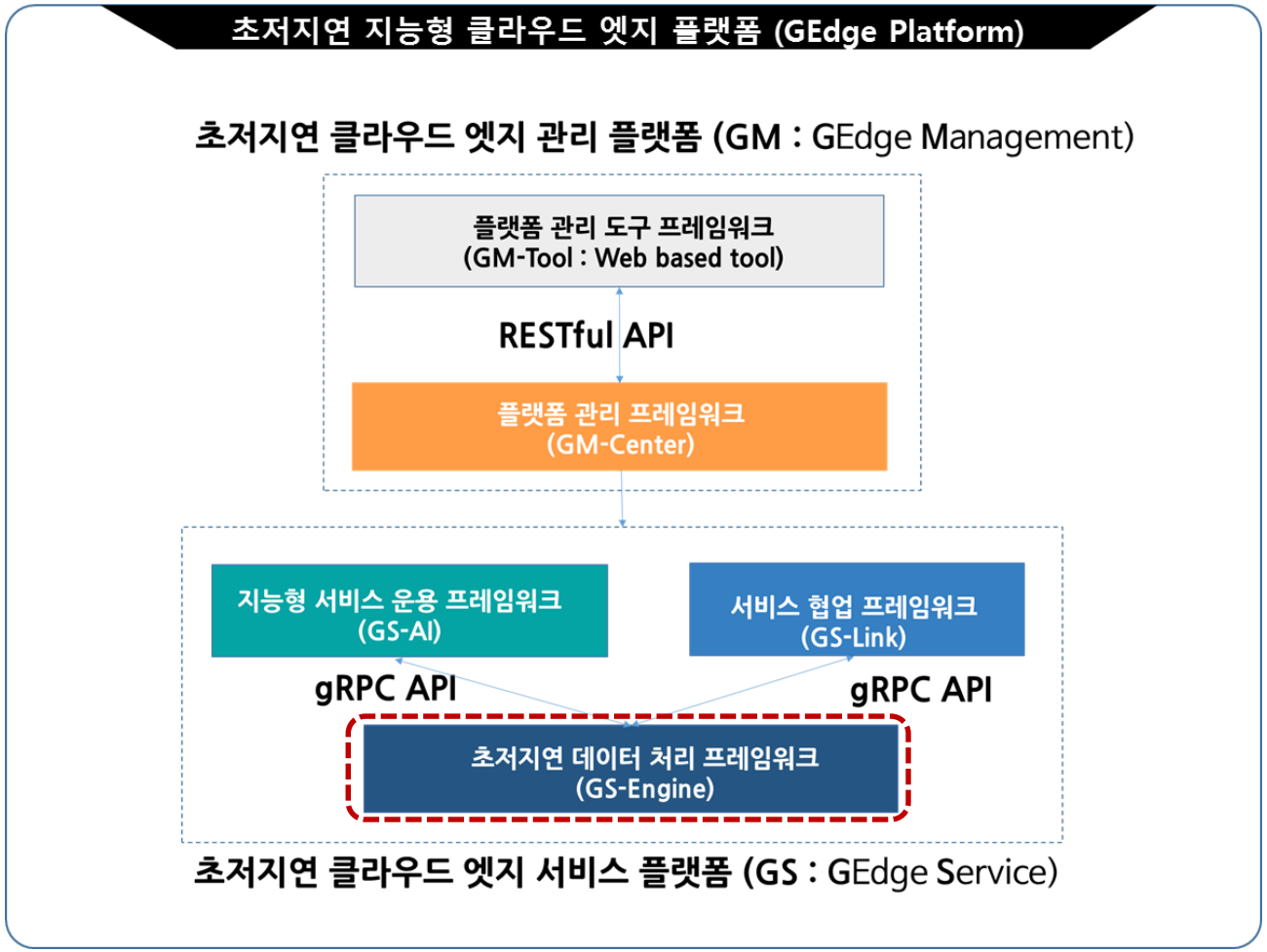
"The First talk of Edge Computing with Clouds"

- GEdge Platform 커뮤니티 멤버들의 첫번째 이야기 -

GEdge Platform Community 1st Conference

# Contents

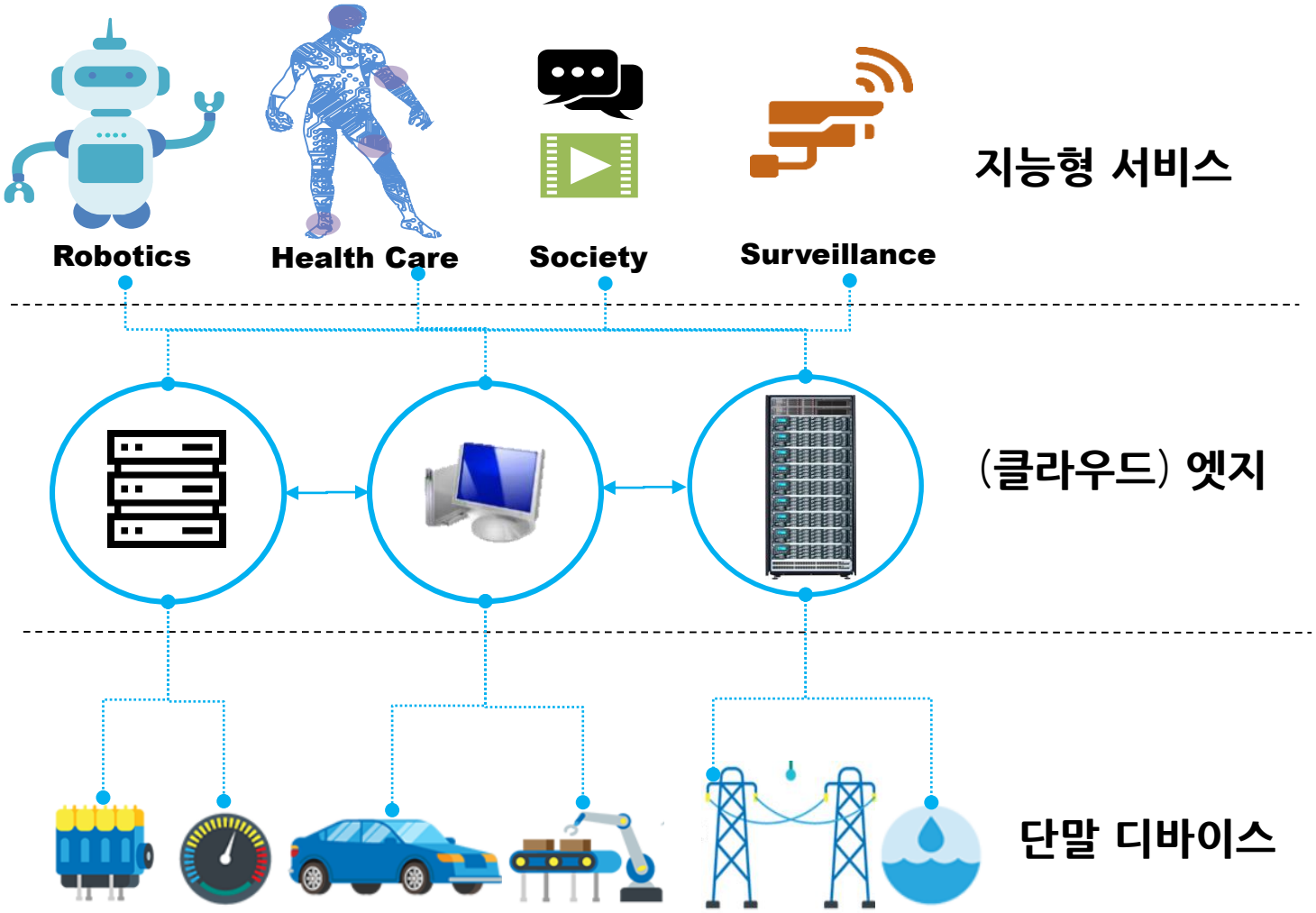
# 이번 세션은 …


초저지연 지능형 클라우드 엣지 플랫폼 (GEdge Platform)
초저지연 클라우드 엣지 관리 플랫폼 (GM : GEdge Management)
플랫폼 관리 도구 프레임워크
(GM-Tool : Web based tool)
RESTful API
플랫폼 관리 프레임워크
(GM-Center)
지능형 서비스 운용 프레임워크
(GS-AI)
서비스 협업 프레임워크
(GS-Link)
gRPC API
gRPC API
초저지연 데이터 처리 프레임워크
(GS-Engine)
초저지연 클라우드 엣지 서비스 플랫폼 (GS : GEdge Service)

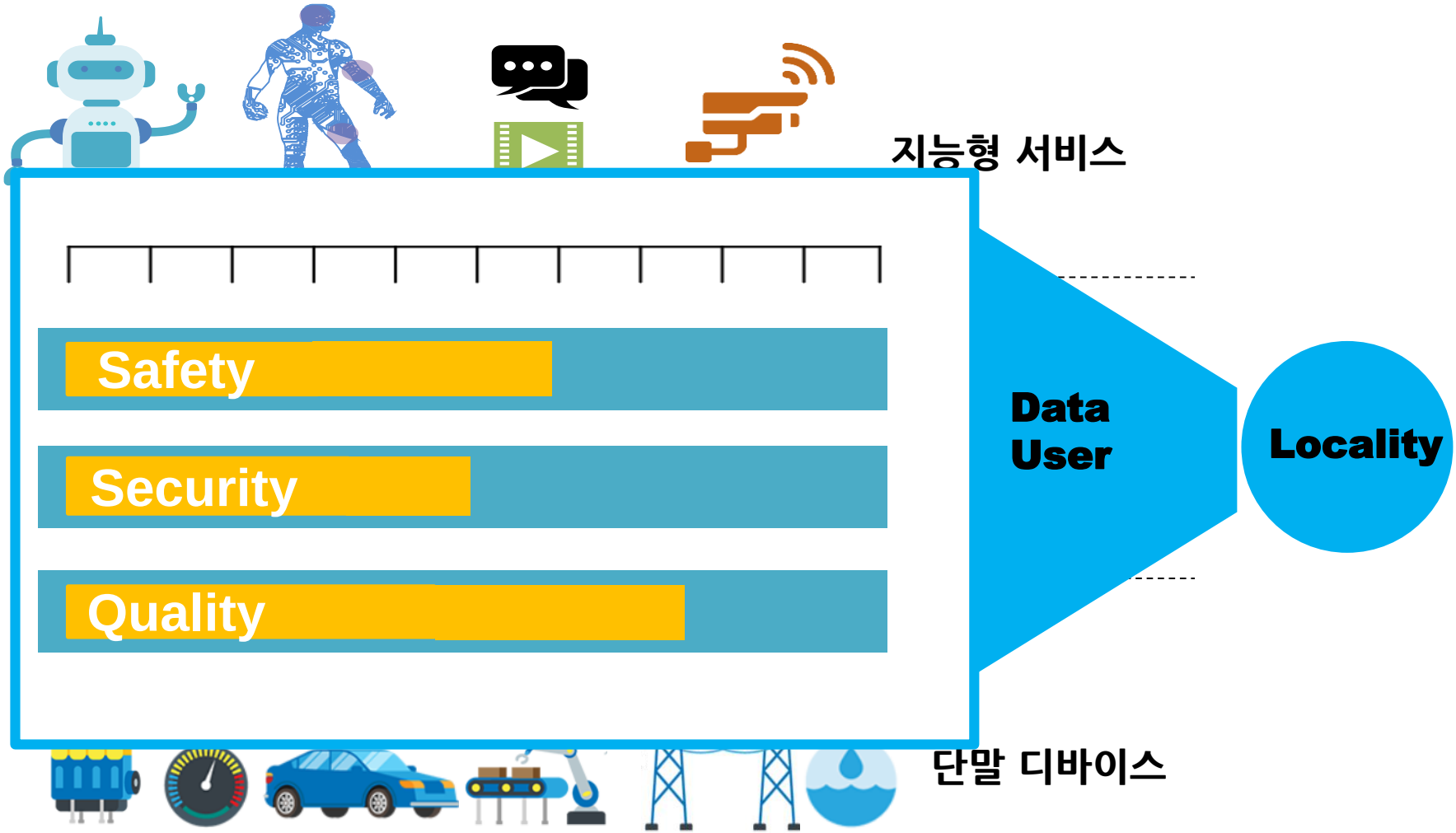
# 1 지능형 엣지 서비스란?

Untact Technology!!

# 1 지능형 엣지 서비스 이점



# 1 지능형 엣지 서비스 이점



# 1 지능형 엣지 서비스 기술 요구사항

**Response Time (Remote Control) : < 1ms**

Smart Port

V2X

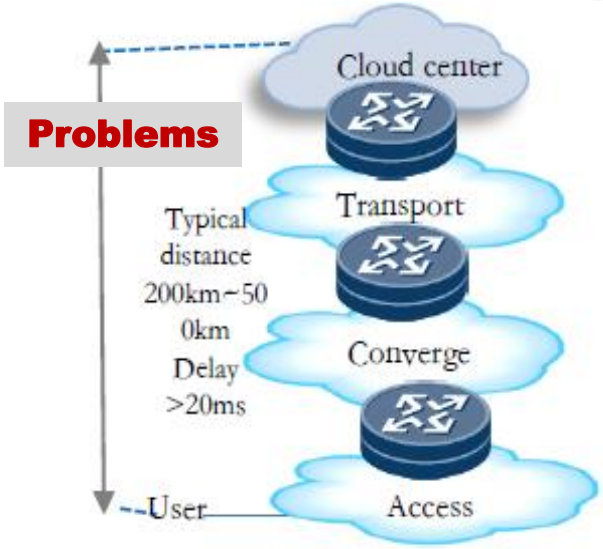
Dron

**Response Time (AR/VR) : < 20ms**

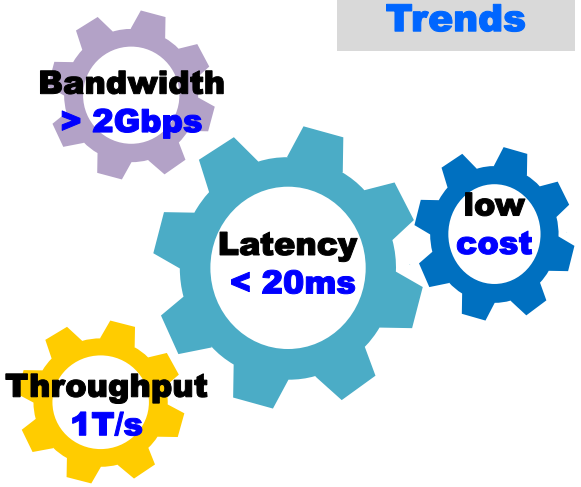
Manufacturing

Exhibition Hall

Telemedicine Diagnosis



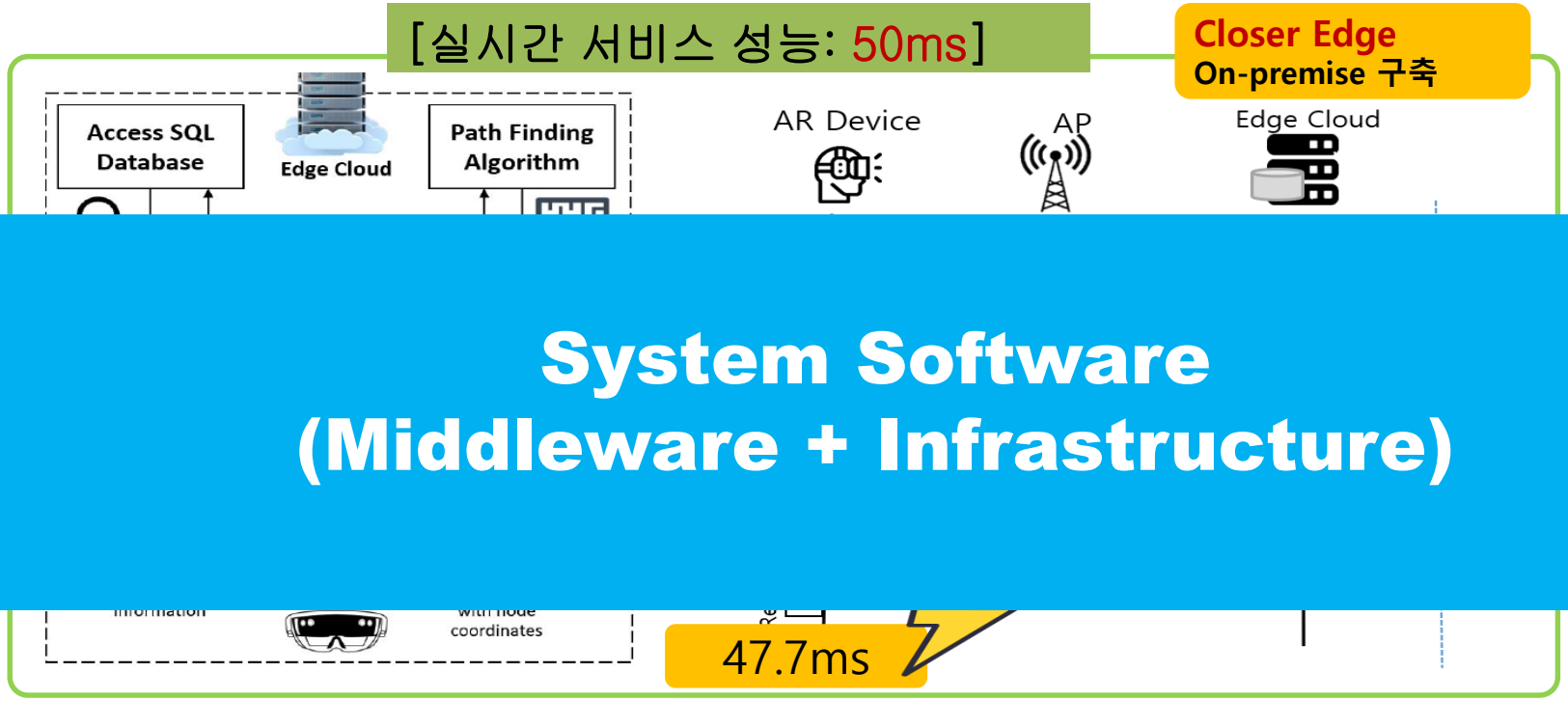

**Trends**




Edge Cloud Infrastructure for the future network (ITU, 2019)

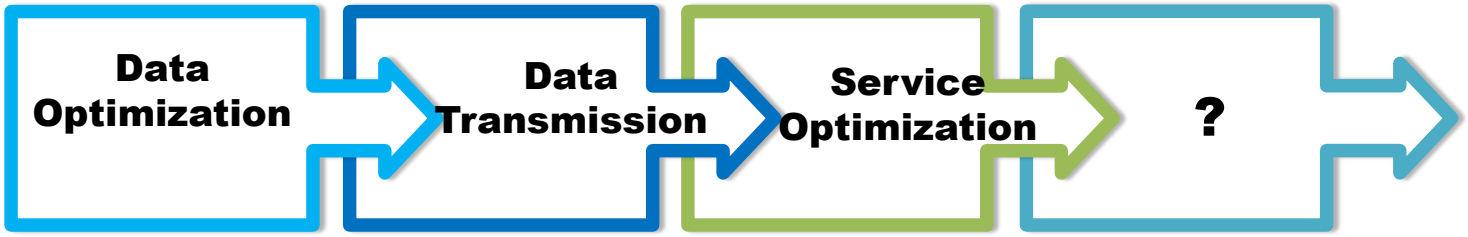
# 1 지능형 엣지 서비스의 기술 현황은?

## Quality: Response Time




**System Software**
**(Middleware + Infrastructure)**

Scalability and Performance Evaluation of Edge Cloud system for latency contained application
(2018, ACM/IEEE Symposium on EC)

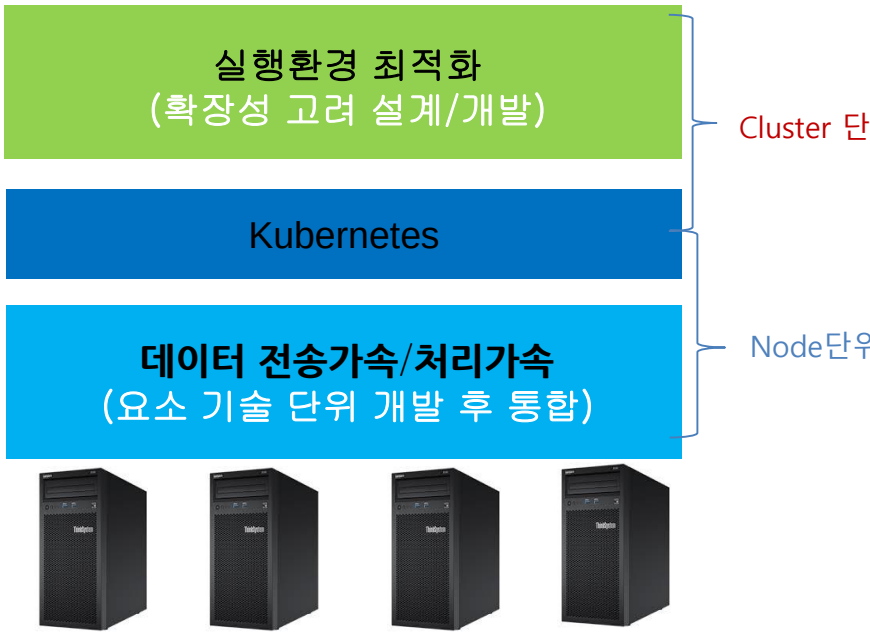


# 2 GS-Engine 소개

GEdge Platform

» 대규모 데이터 기반 엣지 서비스를 제공하기 위하여, **인프라 가속 기술을 바탕으로 서비스 실행환경을 관리**

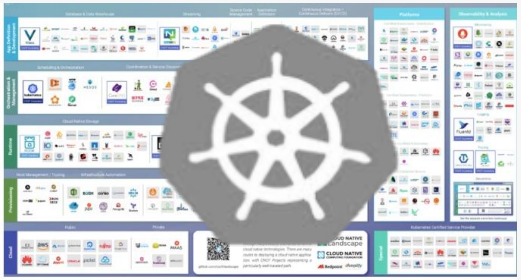
- 엣지 서비스의 **다양한 실행 구조 지원** [Monolithic, Microservices, FaaS]
- 엣지 서비스를 위한 **실행환경의 최적화** [Definition, Launch, Scaling]
- 엣지 서비스의 **성능 가속** [low latency: Processor, Network, Accelerator, Memory]




# 2 GS-Engine 개발 현황

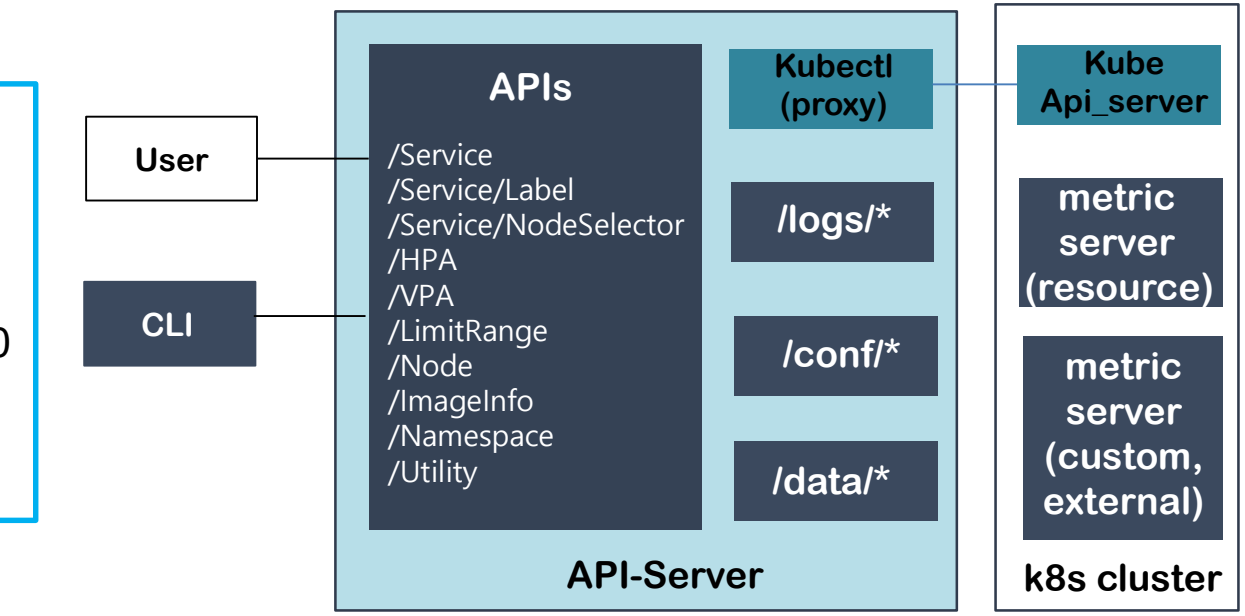
## GS-Engine : API Server

- 빠르고 단순한 엣지 서비스 정의
- 엣지 서비스를 위한 기능 추출/추가/최적화
- Kubernetes의 친절한 상세함/복잡성에서 해방~



### 개발환경

- Ubuntu 18.04 LTS
- Kubernetes v1.18.6
- Docker: 19.03.13
- Nvidia-docker: 1.0.0.-rc10
- Python: 3.7+
- Flask: 1.1.2




GS-Engine: API Server 구조도

### 실행방법

```
gunicorn app:app --bind=x.x.x.x:8888 --daemon --reload
```

# 2 GS-Engine 개발 현황

## GS-Engine : Service Schema

- User: **꼭 정의가 필요한 것만 입력**
- GS-Engine: **엣지 서비스를 위한 기본값 내장** (default value)

### Service Schema

name: str(required=True)
namespace: str(required=True)
containers: list(required=True)
ports: list(num(), required=True)
type: str(required=True)
monitorPorts: list(num(), required=False)
scheduleHint: map(required=False)

### Ex) Service Definition

```
curl -s -X POST -H 'Content-Type: application/yaml' --data '
name: app1
namespace: edge
containers:
- k8s.gcr.io/echoserver:1.10
ports:
- 8080
- 8000
type: LoadBalancer # [LoadBalancer, NodePort, ClusterIP]
monitorPorts: # opt
- 8000
scheduleHint: # opt
  type: func
  preferred: local # [local, cloud, neighbor]
  nodeSelector:
    gpu: gpu1
    ssd: ssd1
' 'http://129.254.202.96:8888/gse/service/create' | jq
```

# 2 GS-Engine 개발 현황

## GS-Engine : AutoScaling

- 현실 생활 대응형 서비스 => 엣지 서비스
- Data Explosion : 서비스 지연/중지 NO!!
- Data Occurrence : 낮은 운영 비용 Yes!!


실행환경 수 = ceil[현재 실행환경 수] * (현재 메트릭 값/원하는 메트릭 값)

### Metric

**Resource: metrics.k8s.io**
**Custom: custom.metrics.k8s.io**
**External: external.metrics.k8s.io**

### Resource Metric Server

**kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml**

☞ Heapster: kubernetes v1.11부터 deprecated.

### candidates

**Microsoft Azure Adaptor**
**Google Stackdriver**
**Datadog Cluster Agent**
**Kube Metrics Adaptor**
**Prometheus Adaptor**

### Custom Metric Server

**helm install prometheus-operator stable/prometheus-operator **
**--namespace monitoring –set**
**prometheus.prometheusSpec.serviceMonitorSelectorNilUsesHelmValues=false**

☞ https://github.com/directxman12/k8s-prometheus-adapter

☞https://github.com/kubernetes/metrics/blob/master/IMPLEMENTATIONS.md#custom-metrics-api

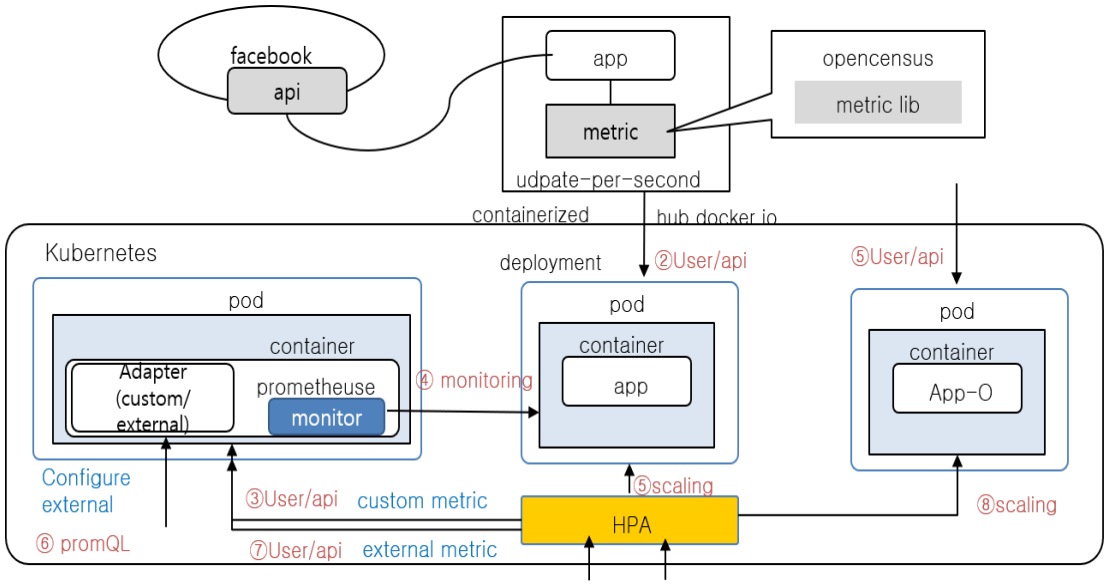
# 2 GS-Engine 개발 현황

## GS-Engine : AutoScaling 지원 방안

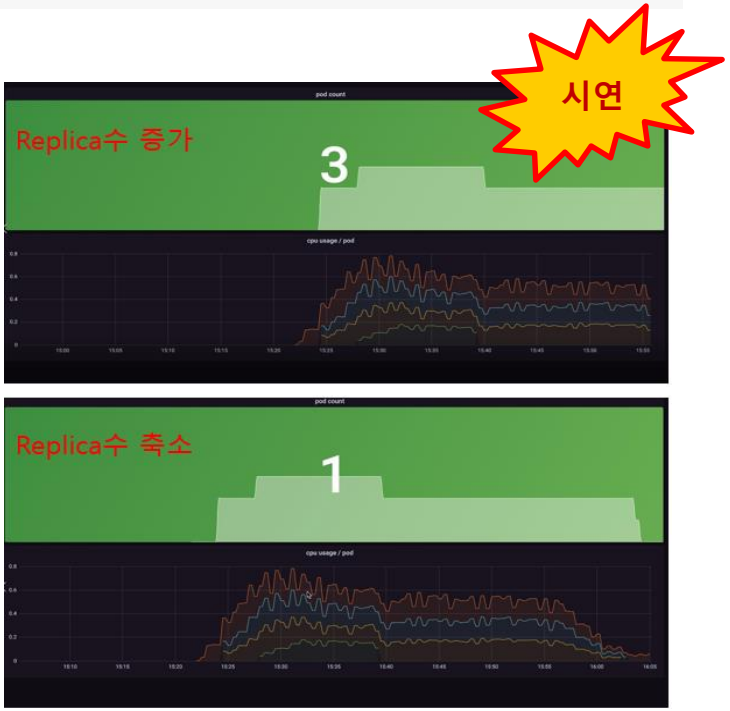
- 1st : Scale out => Scale in
- 2nd : Scale Up => Scale down
- 3rd : over Edge

**Period**

**Scale Out : 3min**
**Scale In: 5min**




Scale Out/In 지원 구조도



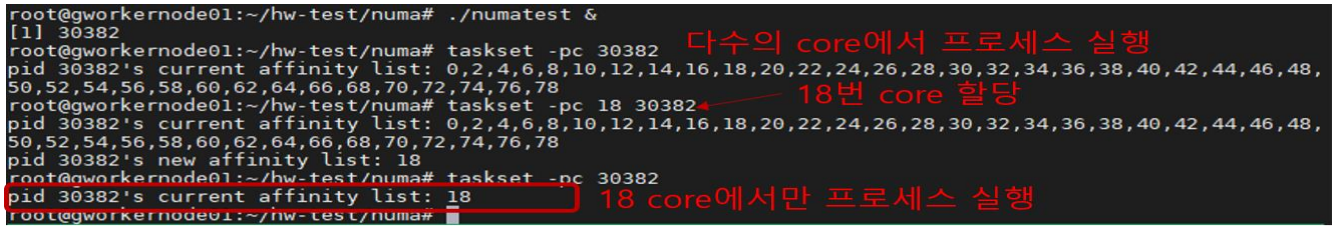

엣지 서비스의 실행환경 수 자동 변이

# 2 GS-Engine 개발 현황

## GS-Engine : 실행 가속 - 자원 선점

- SW 스택: 자원 할당 및 선점 기술
- Container 기반 엣지 서비스: **자원 선점 을 통한 성능 가속**

| SW스택 | 자원 할당 기술 | 정책 |
|---|---|---|
| Linux(OS) | taskset, cset,<br>numactl | - 프로세서의 특정 코어 할당 지원<br>- core의 로컬메모리 우선 할당 |
| Docker | docker run 의 옵션<br>--memory, --memroy-swapiness<br>--cpus, --cpuset-cpus<br>--gpus | - memory와 스왑에 사용 설정<br>- CPU와 kernel 스케줄러 선택 지원<br>- 특정 GPU선정 |
| Kubernetes | CPU manager, Topology manager 기반 자원 관리 및 선점 스케줄링 개발 | - kubernetes v1.18부터 default 지원<br>- Guaranteed class 기반 pod 실행 시 static 정책하에서 cpu 자원의 선정 가능<br>- NVIDIA GPU device plugin, Intel SRIOV network device Plugin 등 연동 기반 개발 진행중 |




**프로세서 선점**

# 2 GS-Engine 개발 현황

## GS-Engine : 성능 가속 - 자원 선점

- CPU **자원 선점 실험** : **52% 성능 향상**
- **10코어를 대상으로 동영상 인코딩 응용 실행**: 2252초
- 10코어를 선점 기반 동영상 인코딩 응용 실행: 1290초

/var/lib/kubelet/config.yaml

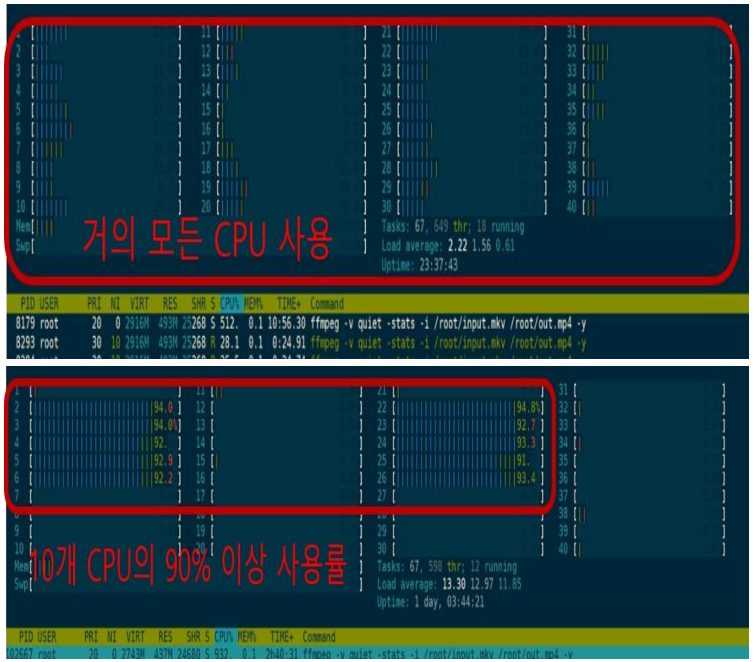
```
apiVersion: kubelet.config.k8s.io/v1beta1
authentication:
  anonymous:
    enabled: false
...

clusterDomain: k8s.local
cpuManagerReconcilePeriod: 10s
cpuManagerPolicy: static
cpuFSQuotaPeriod: 100ms
 ….
kubeReserved:
  cpu: 500m
```

cpu-manger-test.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: cpu-manager-test
  namespace: default
spec:
  containers:
  - name: cpu-manager-test
    image: alicek106/stress
    args: ["tail", "-f", "/dev/null"]
    resources:
     limits:
      cpu: 1
      memory: 200Mi
```

엣지 서비스의 CPU 선점




Container **기반 엣지 서비스 실행**

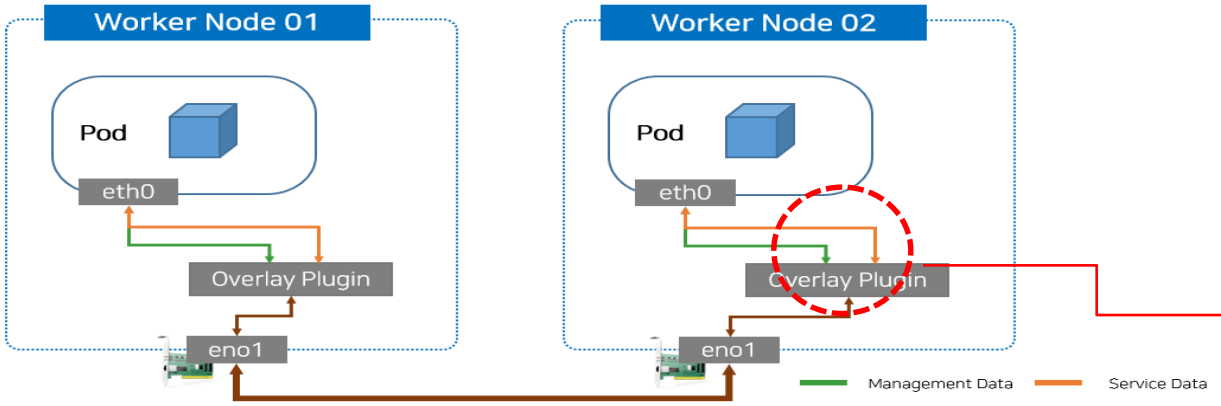
```
docker run -d --name cpuset_2  --cpuset-cpus=2 alicek106/stress stress --cpu 1
```

# 2 GS-Engine 개발 현황

## GS-Engine : 성능 가속 - 네트워크

- **네트워크 공유**: 클러스터 관리 vs 엣지 서비스
- **엣지 서비스 공유**: 멀티 응용간 데이터 전송 패스 이원화
- CNI을 지원하는 다양한 network project 들의 최적 활용

ROMANA cilium MULTUS flannel weavenet canal PROJECT CALICO KUBE ROUTER




기본 엣지 서비스를 위한 네트워크 구성도

**엣지서비스 가속**

**전송 가속 엣지 서비스**

⇒ 대역폭 우선권

⇒ HW 기반 네트워크 가상화

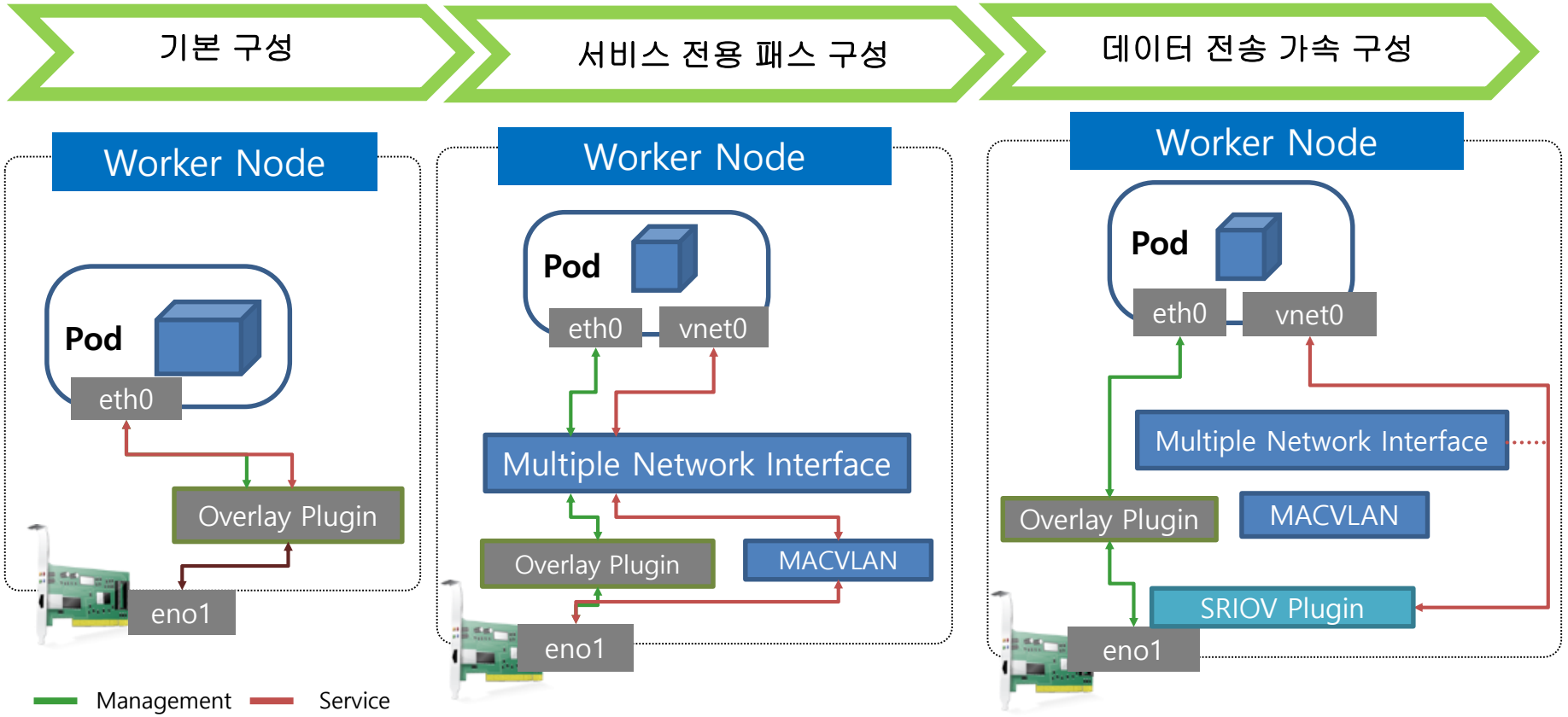
**Problem**

하나의 네트워크가
다른 목적으로 공유됨

# 2 GS-Engine 개발 현황

## GS-Engine : 성능 가속 - 네트워크

- 다양한 네트워크 장치 구성
- 엣지 서비스의 전송 성능 실험: 데이터 전송 대역폭(시간) 비교

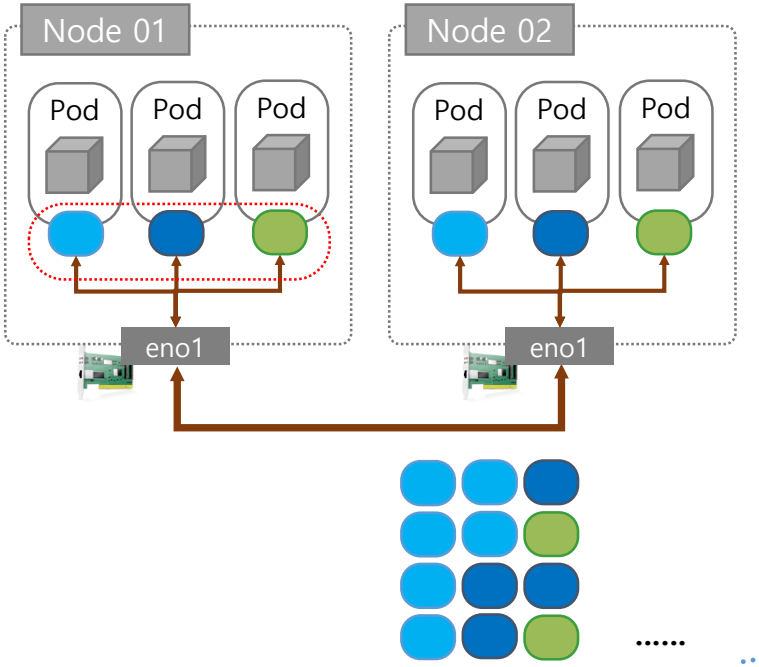


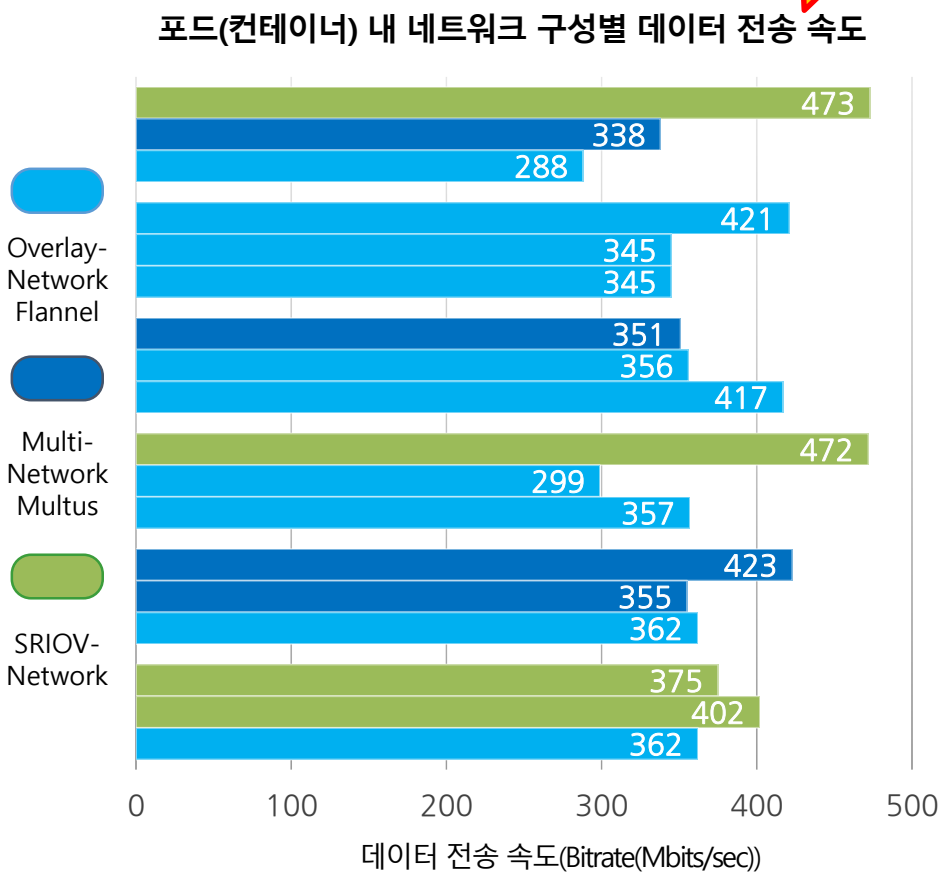
# 2 GS-Engine 개발 현황

## GS-Engine : 성능 가속 - 네트워크

- **엣지 서비스 전송 가속**: Multus(**패스 이원화**) + SRIOV(HW **가속**)

시연



# 2 향후 개발 계획

GEdge Platform

**2020**

실행환경

- Monolithic 지원
- Service Schema
- Scaling(Out/In)
- 다중 메트릭 지원 (Resource, Custom)

Infra

- 자원선점(CPU)
- 다중 네트워크 지원
- 네트워크 패스이원화

**2021**

실행환경

- Microservices 지원
- Service Schema 확장
- Scaling(Up/Down)
- 다중 메트릭 확장 (External)

Infra

- 자원 선점확장(CPU)
- 네트워크 필터링
- 가속 자원 지원

**2022**

실행환경

- Function 지원
- Service Schema 확장
- Scaling(over Edge)
- 이벤트 관리기

Infra

- 자원 선점 (HW)
- 네트워크 필터링
- RDMA
- 네트워크 라우팅패스 구성

**2023**

실행환경

- 다중 실행환경 최적화
- 이벤트 기반 서비스 실행기

Infra

- 자원 선점확장(HW)
- 네트워크 라우팅패스 확장 및 구동
- 5G 모바일 엣지 네트워크 연동

# 3 기술 데모


AutoScaling
Scaling Out/In
With Custom Metrics
실행환경
Network
Network bandwidth
Comparison
성능 가속

# 감사합니다.

http://gedge-platform.github.io

GS-Engine 코어 개발자
최현화(hyuwnha@etri.re.kr)