최적 자원 배치를 위한 스케줄링

(GS-Scheduler)

2020.12.10

GEdge Platform 코어 개발자
장수민(jsm@etri.re.kr)

"The First talk of Edge Computing with Clouds"

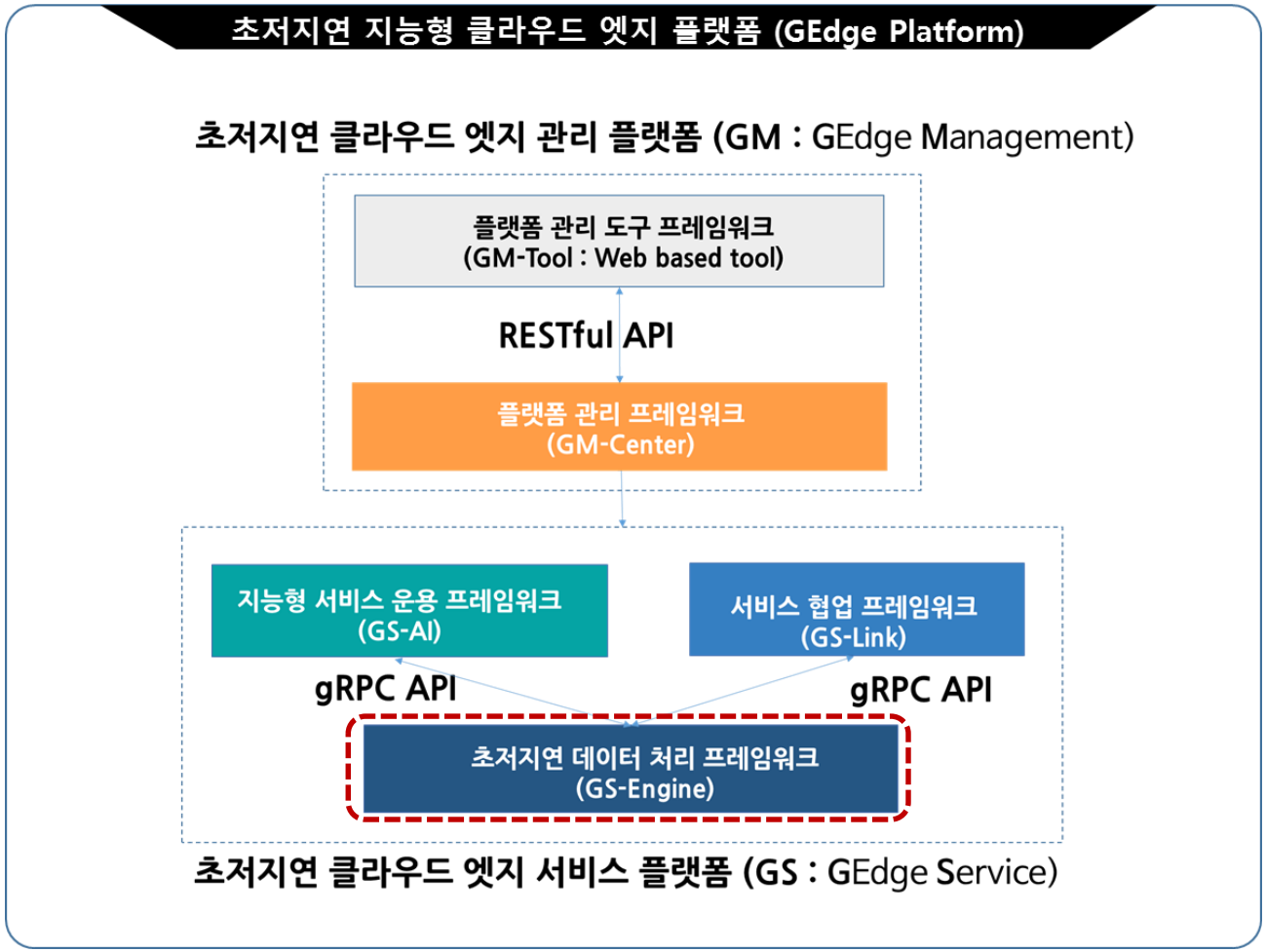
- GEdge Platform 커뮤니티 멤버들의 첫번째 이야기 -

GEdge Platform Community 1st Conference

Contents

이번 세션은 …

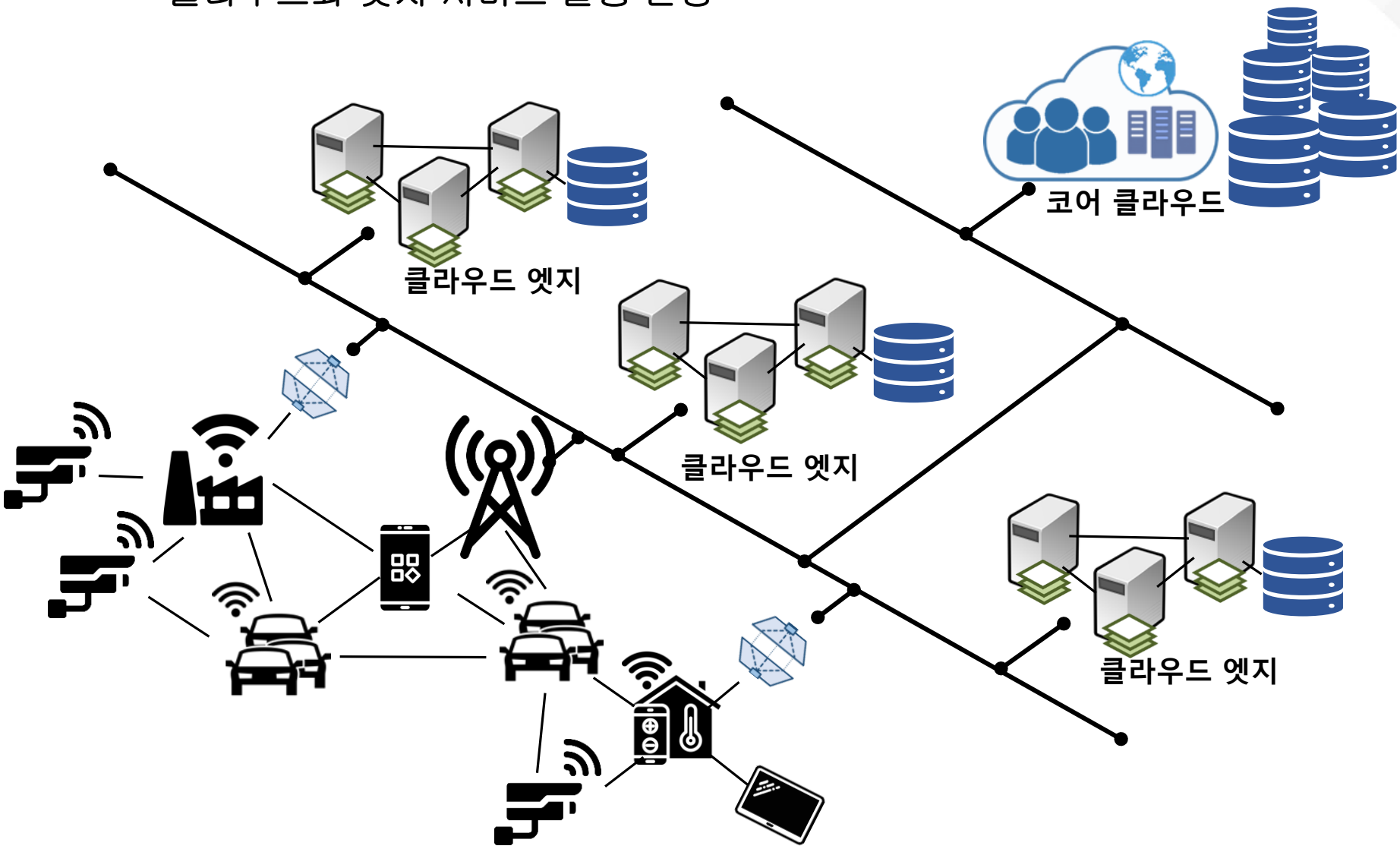
이 장표를 넣어 주시고 해당 그림은 컨퍼런스 발표 내용에 맞게 수정하여 재배포하겠습니다.



GS-Scheduler의 개요

1 GS-Scheduler의 개요

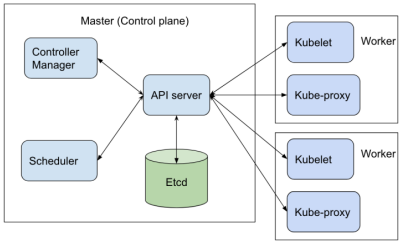
클라우드와 엣지 서비스 실행 환경



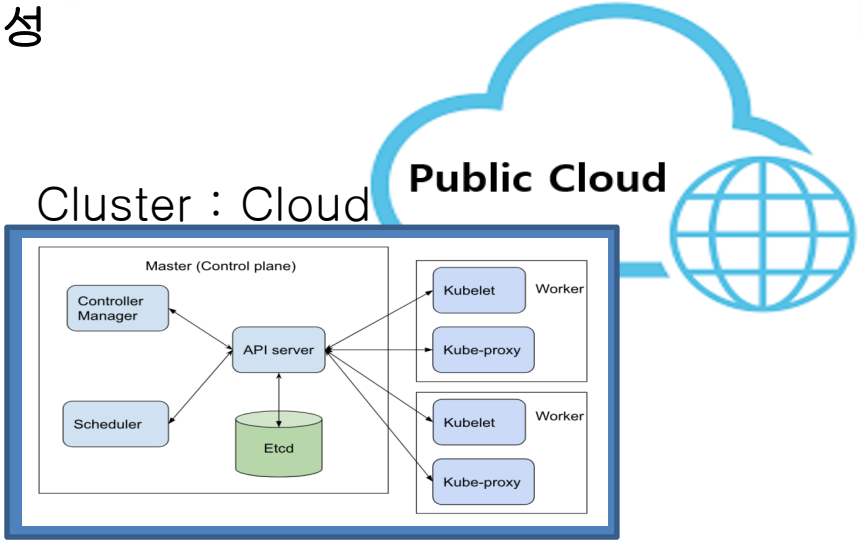
1 GS-Scheduler의 개요

클라우드와 엣지 서비스 시스템 구성

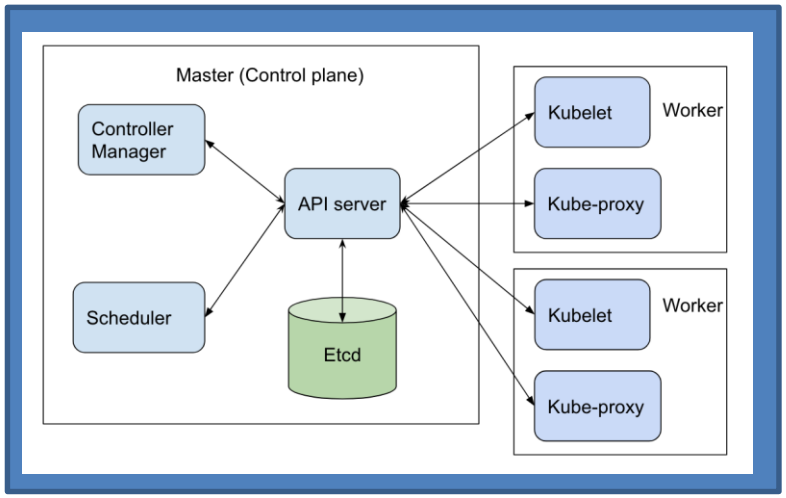
Cluster: Master Cluster



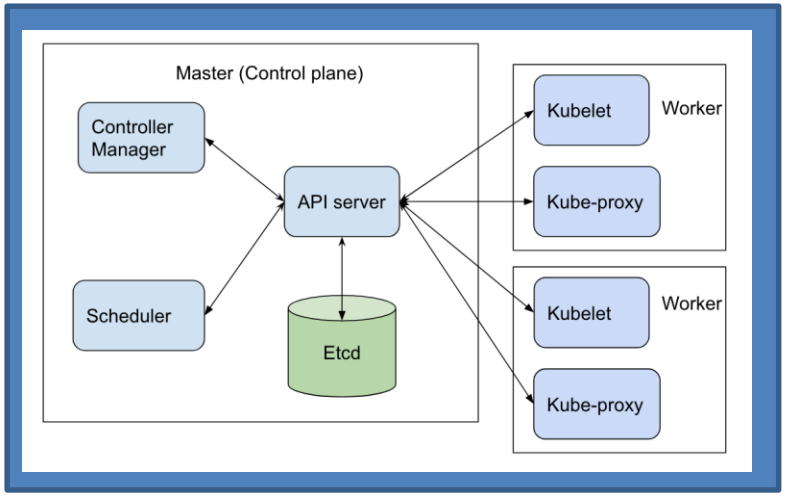
Cluster : Cloud



Cluster : Edge

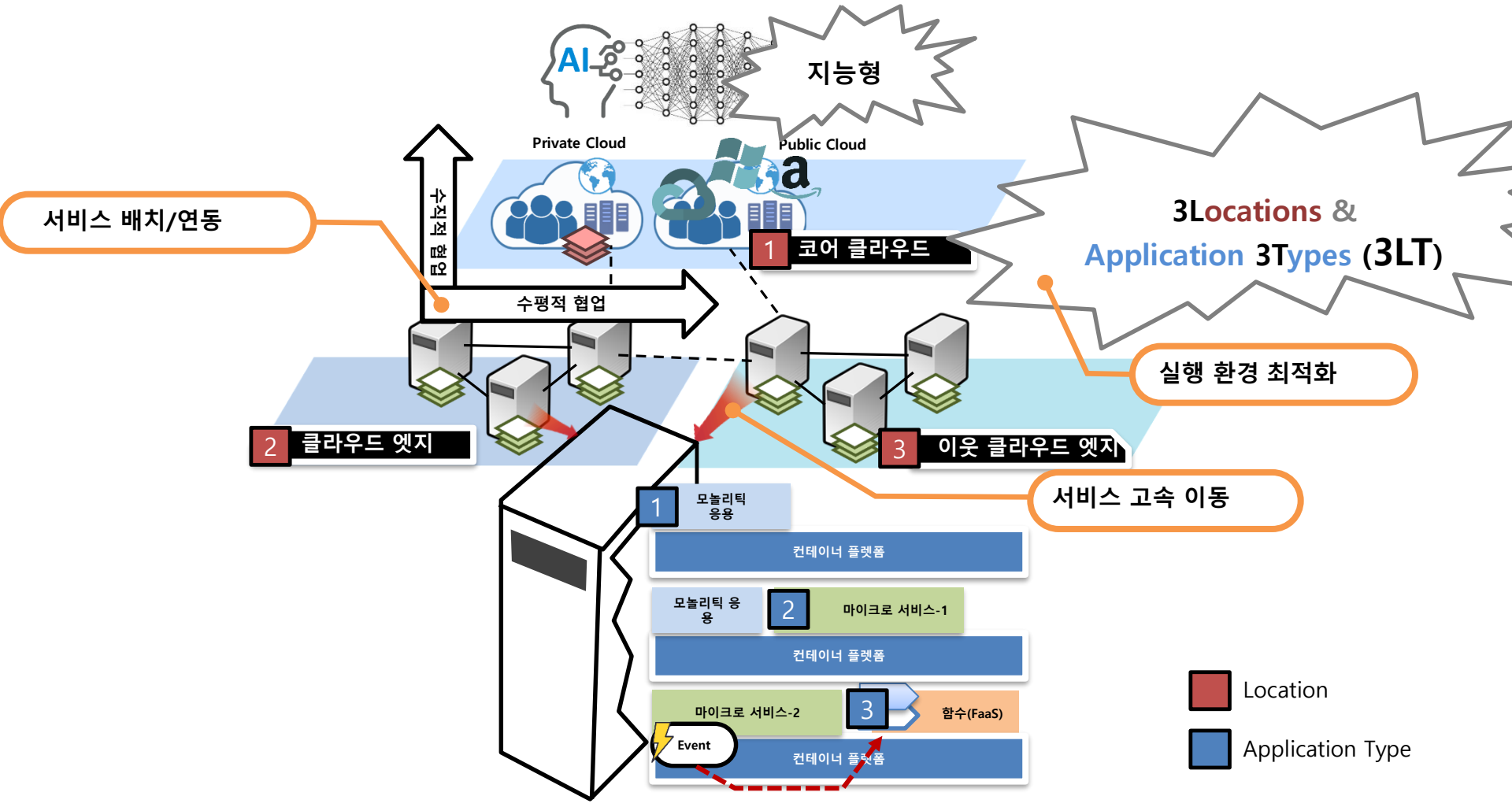


Cluster : Edge(Near Edge)



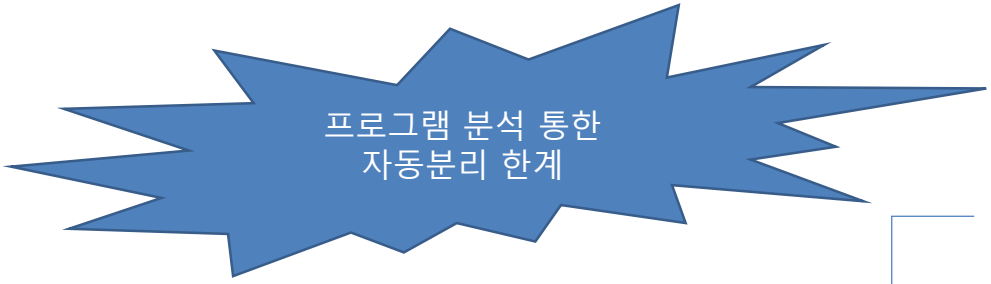
1 GS-Scheduler의 개요

3LT/수직·수평적 서비스 협업



1 GS-Scheduler의 개요

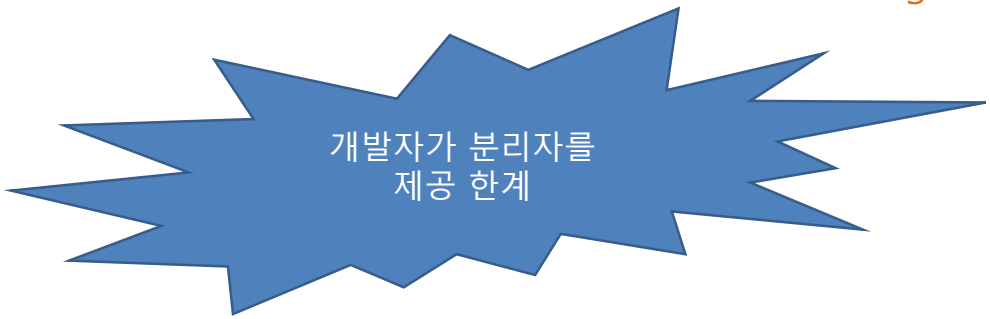
클라우드와 엣지 서비스-협업 이슈



Challenges in Computing Offloading

- Automatic Program Analysis
- Programmer-Specified
- Multi-node Partitioning
- Partitioning Granularity

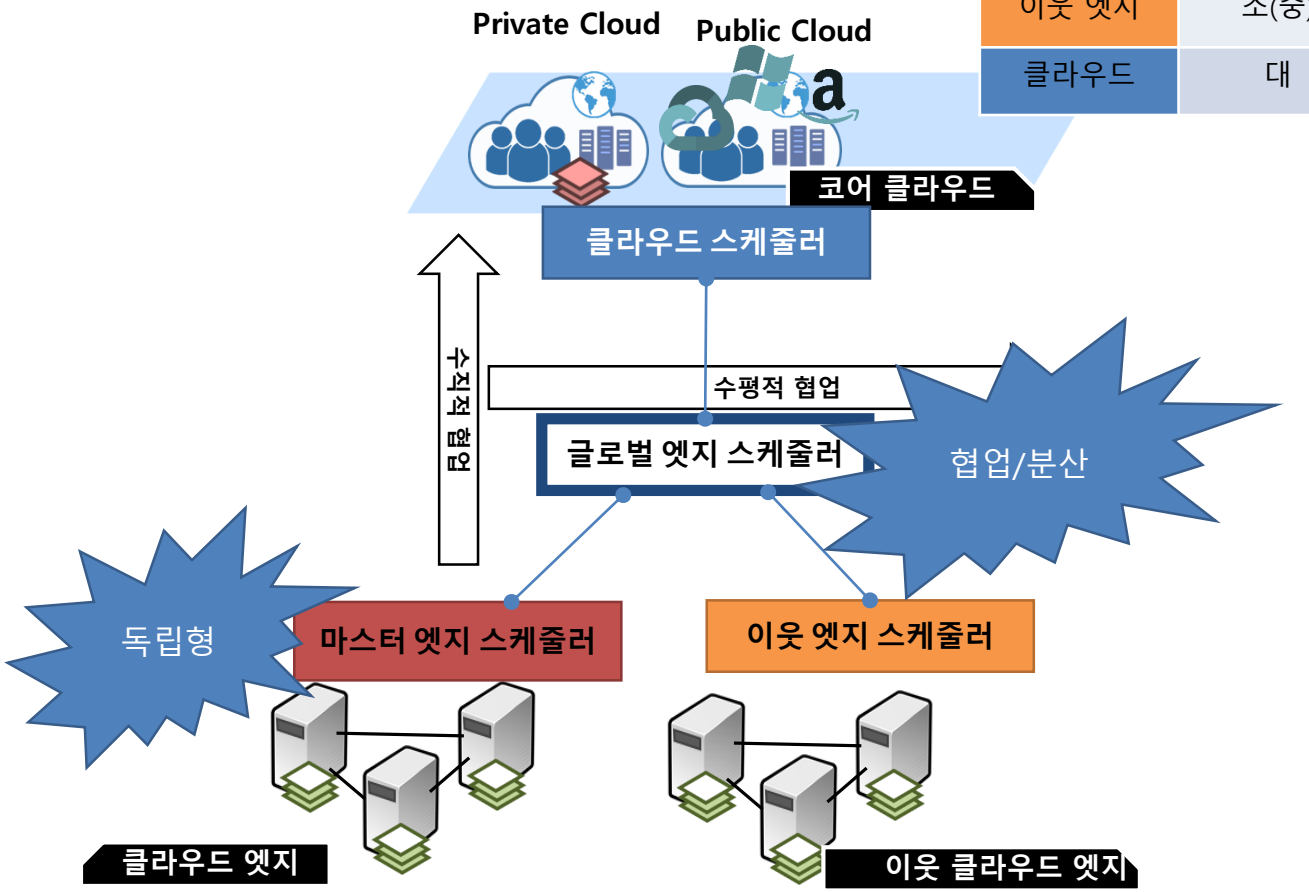
- Task Allocation
- Task Scheduling
- Resource Management



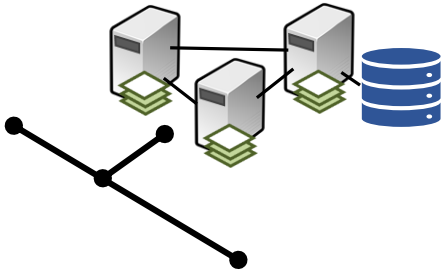
1 GS-Scheduler의 개요

GS-Scheduler 개념도

구분	리소스	네트워크	연산처리
마스터 엣지	소(중)	상	중(소)
이웃 엣지	소(중)	중	중(소)
클라우드	대	하	상

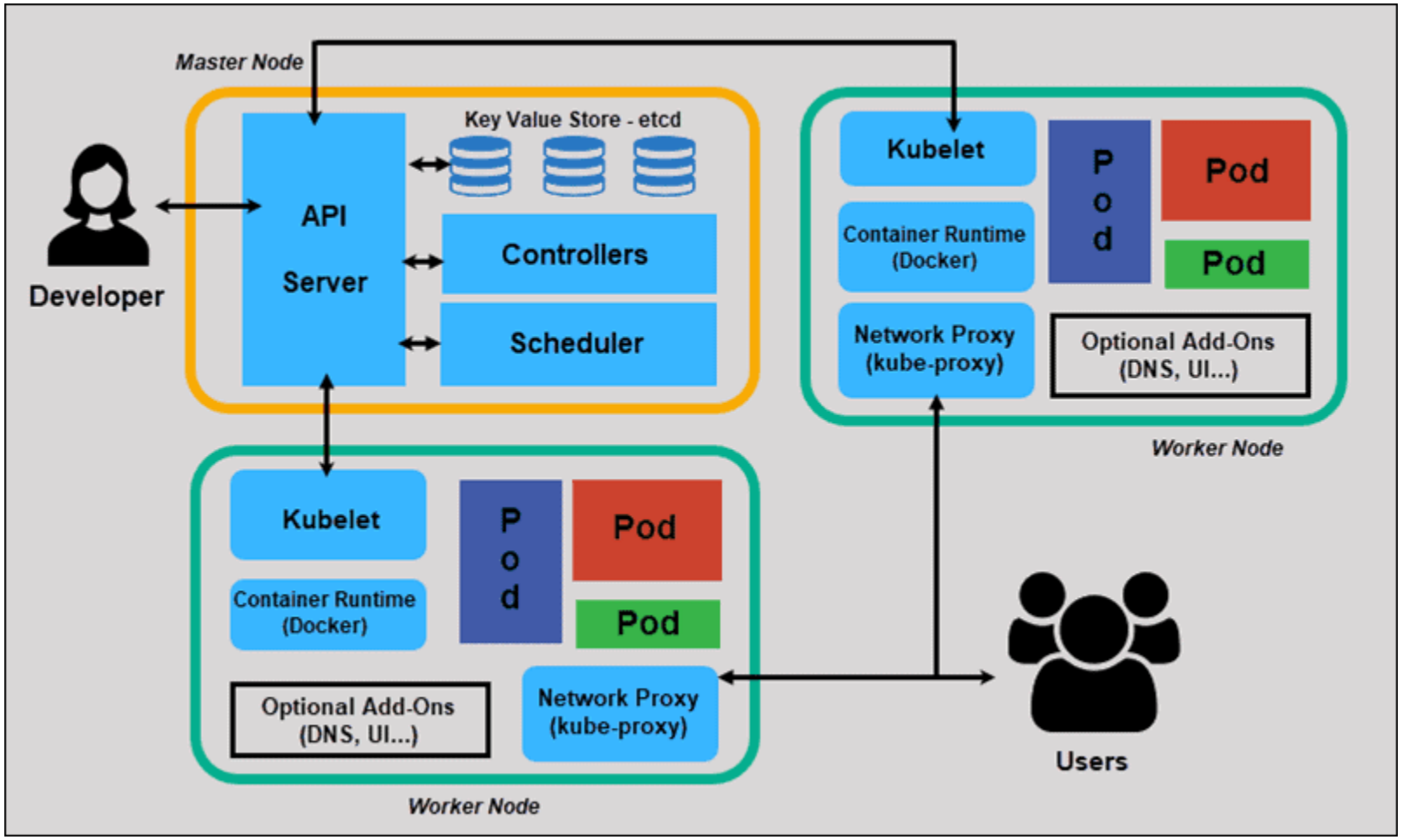


Kubernetes-Custom-Scheduler



2 Kubernetes-Custom-Scheduler

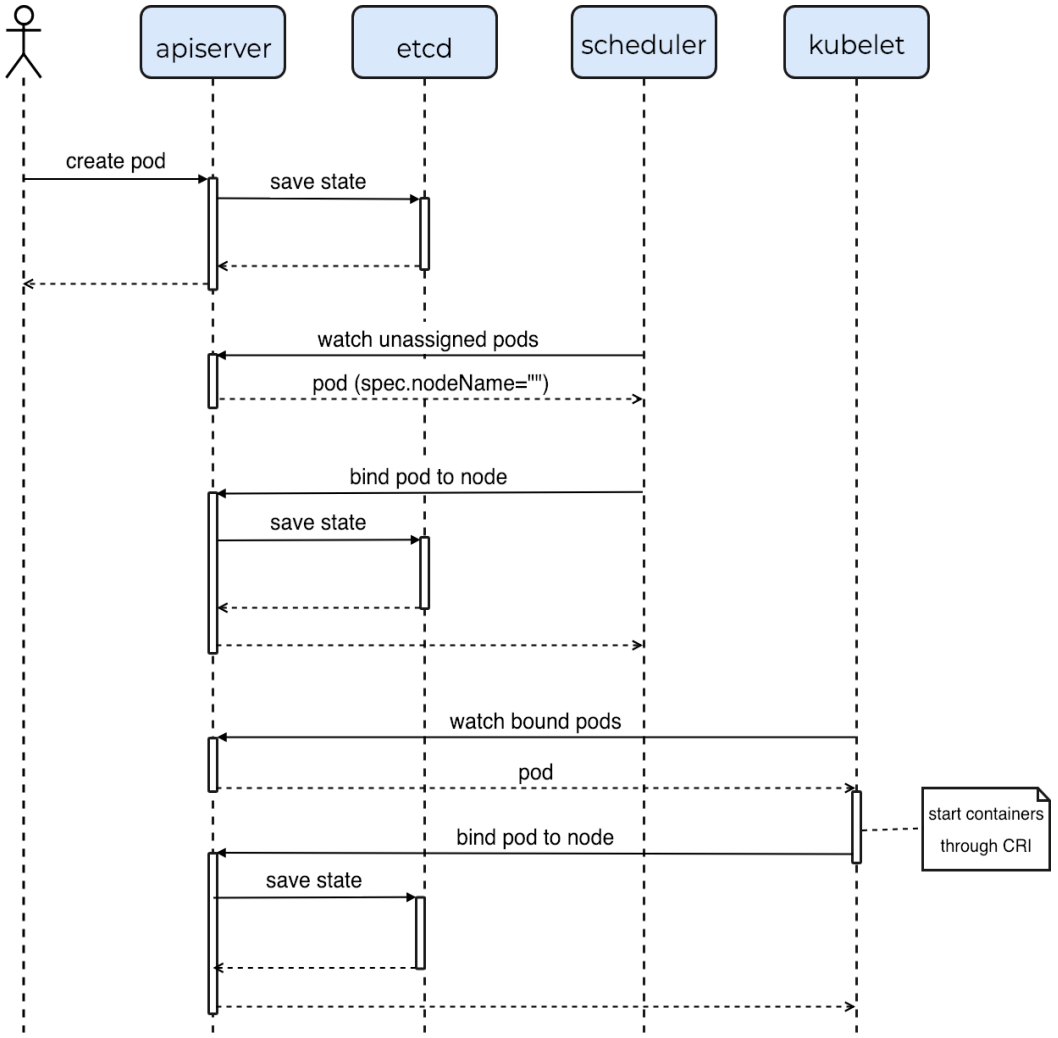
Kubernetes 구성도



2 Kubernetes-Custom-Scheduler

Kubernetes-Custom-Scheduler 처리 절차

1. kubectl apply 등의 명령어를 통해 사용자의 Pod의 생성 요청이 kube-apiserver에 제출되면 kube-apiserver는 etcd에 Pod의 정보를 저장한다.
2. Pod가 위치한 노드의 정보인 NodeName의 값이 설정되지 않은 상태로 Pod의 정보를 저장한다.
3. kube-scheduler는 이러한 정보를 watch하고 있다가 Pod의 NodeName이 비어 있는 상태라는 것을 감지한다.
4. 해당 Pod를 할당하기 위한 적절한 노드에 찾는, 일종의 스케줄링 작업을 진행한다.
5. 적절한 노드를 찾았다면, kube-scheduler는 그 정보를 kube-apiserver에게 전달한다.
6. Pod를 직접 생성은 kubelet에서 처리한다.
7. Pod의 상태 정보를 업데이트 한다.



2 Kubernetes-Custom-Scheduler

GEdge Platform

Kubernetes-Custom-Scheduler 예시

Configure your pods to use a custom Kubernetes scheduler

First, you need to configure your pods to use a custom scheduler:

```yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
    app: nginx
  template:
    metadata:
      name: nginx
      labels:
        app: nginx
    spec:
      schedulerName: sysdigsched
      containers:
      - name: nginx
        image: nginx
        ports:
        - name: http
          containerPort: 80
```

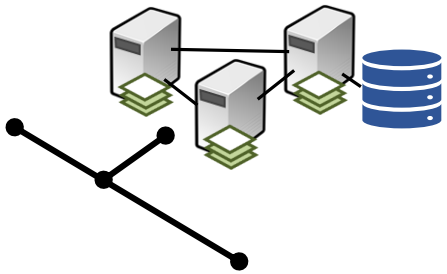
```
$ kubectl create -f nginxrc.yaml
replicationcontroller "nginx" created

$ kubectl get pods
NAME                    READY     STATUS    RESTARTS   AGE
nginx-84cnn             0/1       Pending   0          11s
nginx-ff1dk             0/1       Pending   0          11s
nginx-jq5jk             0/1       Pending   0          11s
```

2 Kubernetes-Custom-Scheduler

Kubernetes-Custom-Scheduler Sample Code

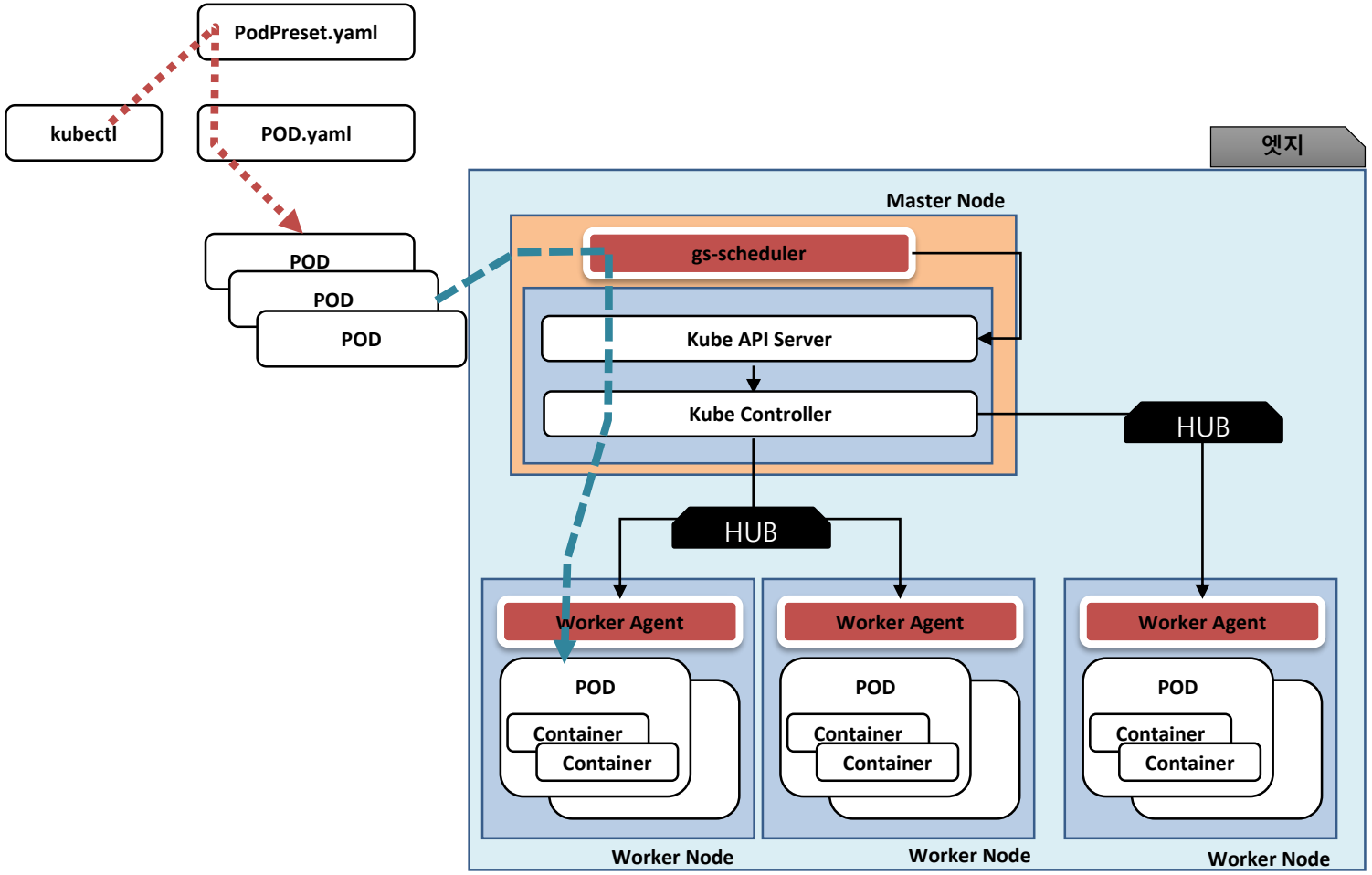
```python
import time
import random
import json

from kubernetes import client, config, watch

config.load_kube_config()
v1=client.CoreV1Api()
scheduler_name = "foobar"

def nodes_available():
    ready_nodes = []
    for n in v1.list_node().items:
        for status in n.status.conditions:
            if status.status == "True" and status.type == "Ready":
                ready_nodes.append(n.metadata.name)
    print(ready_nodes)
    return ready_nodes

def scheduler(name, node, namespace="default"):
    target = client.V1ObjectReference(kind = 'Node', api_version = 'v1', name = node)
    meta = client.V1ObjectMeta(name = name)
    body = client.V1Binding(target = target, metadata = meta)
    try:
        client.CoreV1Api().create_namespaced_binding(namespace=namespace, body=body)
    except ValueError:
        print(1)

def main():
    w = watch.Watch()
    for event in w.stream(v1.list_namespaced_pod, "default"):
        if event['object'].status.phase == "Pending" and event['object'].spec.scheduler_name == scheduler_name:
            try:
                print(event['object'].metadata.name)
                scheduler(event['object'].metadata.name,random.choice(nodes_available()))
            except client.rest.ApiException as e:
                print (json.loads(e.body)['message'])

if __name__ == '__main__':
    main()
```

클라우드 엣지 컴퓨팅을 위한 전용 로컬 스케줄러 개발



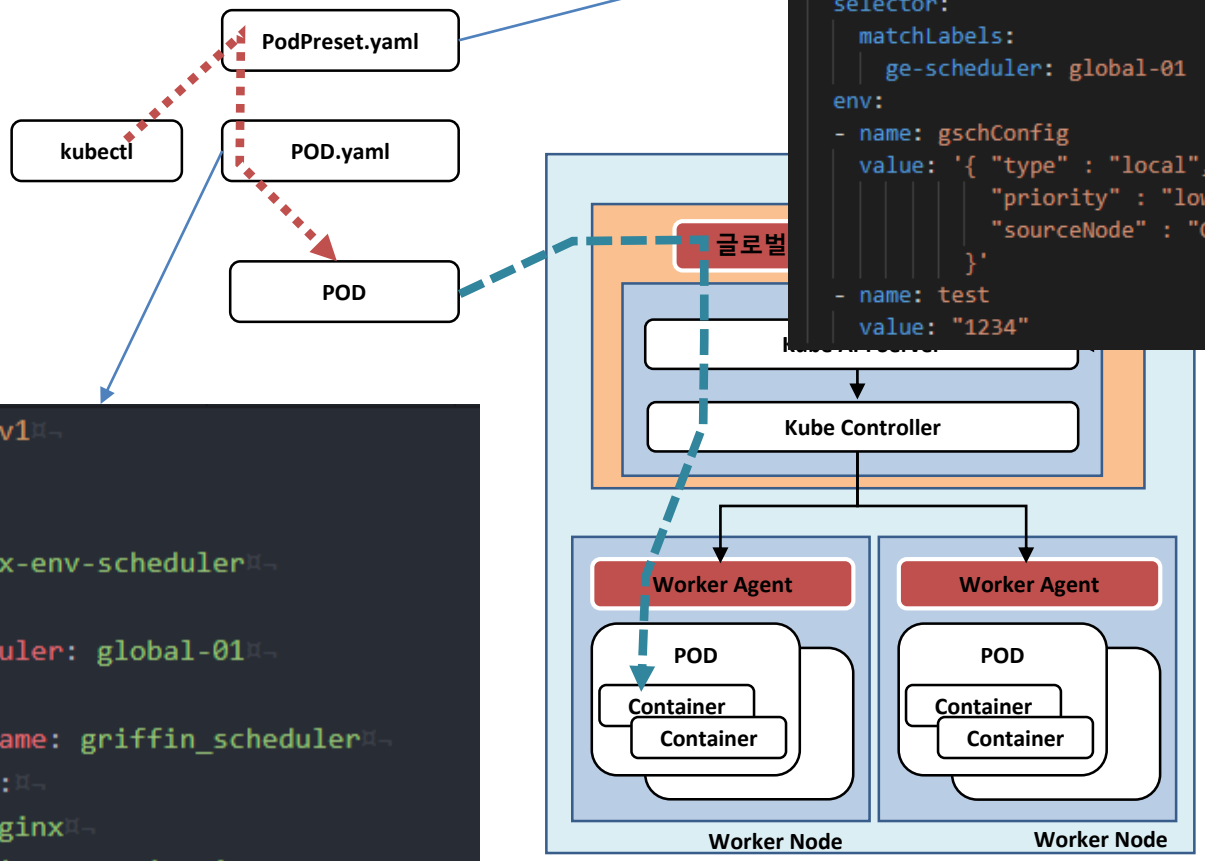
3 클라우드 엣지 컴퓨팅을 위한 전용 로컬 스케줄러 개발

GS-Scheduler 처리 개념도



3 클라우드 엣지 컴퓨팅을 위한 전용 로컬 스케줄러 개발

GS-Scheduler 처리 개념도

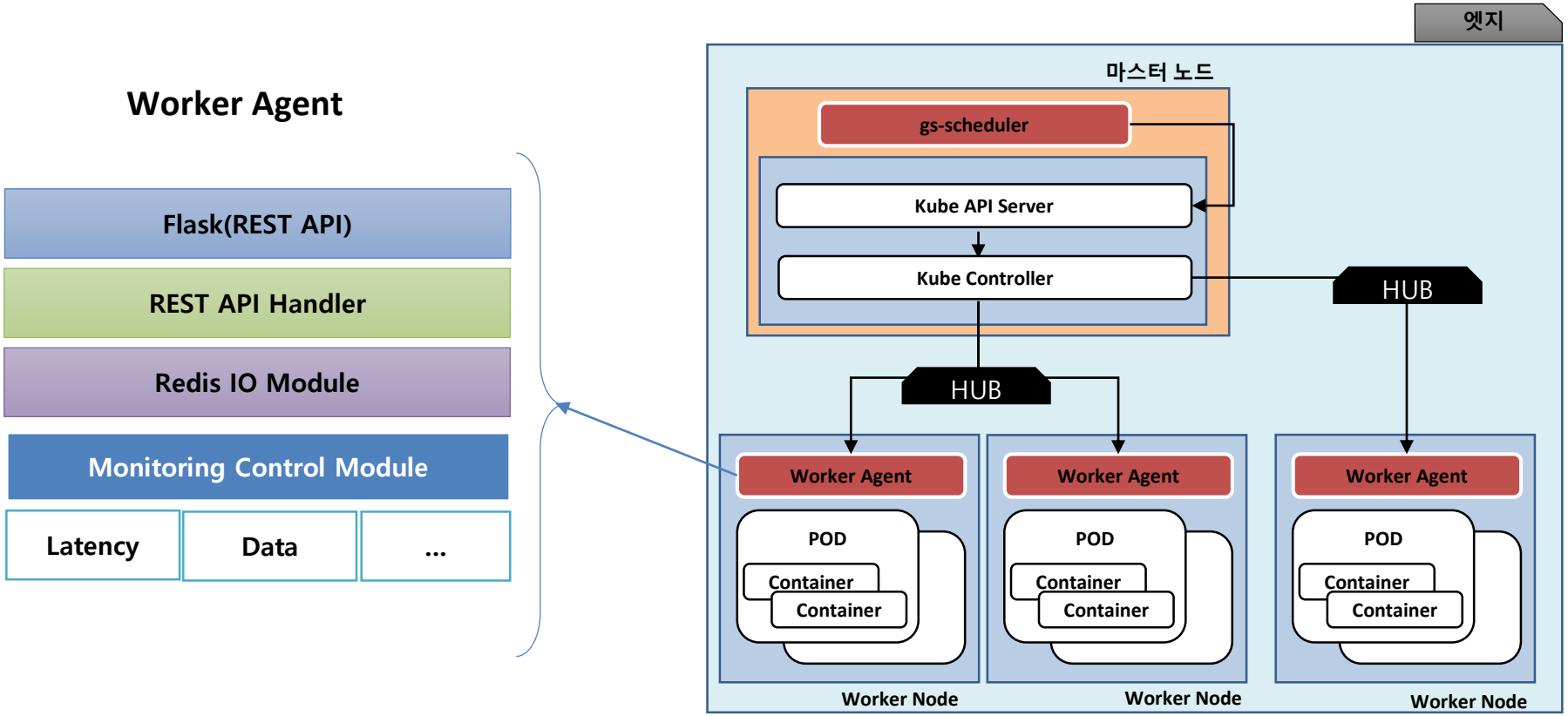



```yaml
apiVersion: settings.k8s.io/v1alpha1
kind: PodPreset
metadata:
  name: ge-podpreset-global-01
spec:
  selector:
    matchLabels:
      ge-scheduler: global-01
  env:
  - name: gschConfig
    value: '{ "type" : "local",
              "priority" : "low-latency",
              "sourceNode" : "GEdge01-master"
            }'
  - name: test
    value: "1234"
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-env-scheduler
  labels:
    ge-scheduler: global-01
spec:
  schedulerName: griffin_scheduler
  containers:
  - image: nginx
    name: nginx-container1
```

클라우드 엣지 컴퓨팅을 위한 전용 로컬 스케줄러 개발

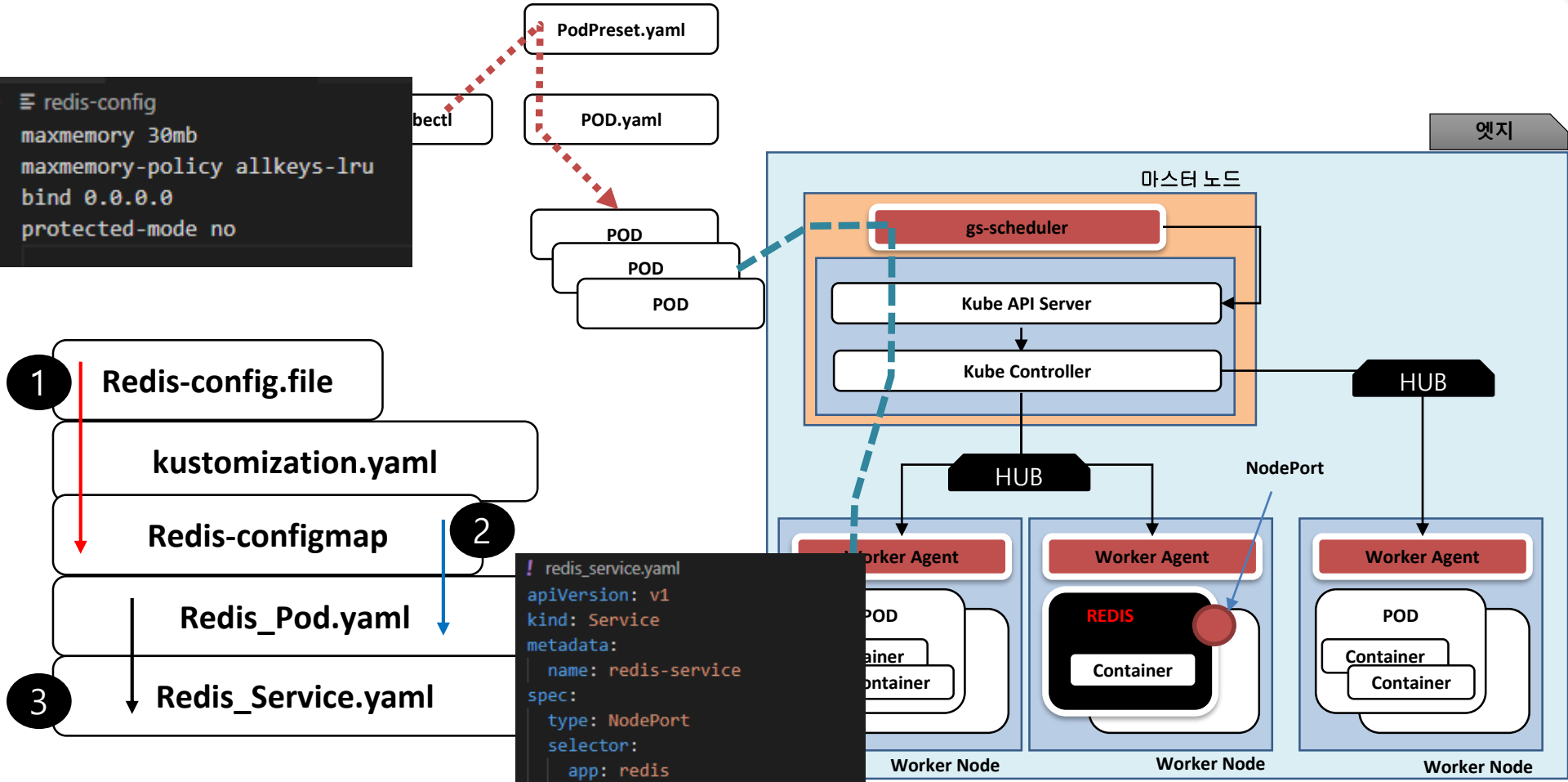
GS-Scheduler 의 Worker Agent



3 클라우드 엣지 컴퓨팅을 위한 전용 로컬 스케줄러 개발

GS-Scheduler 의 Redis

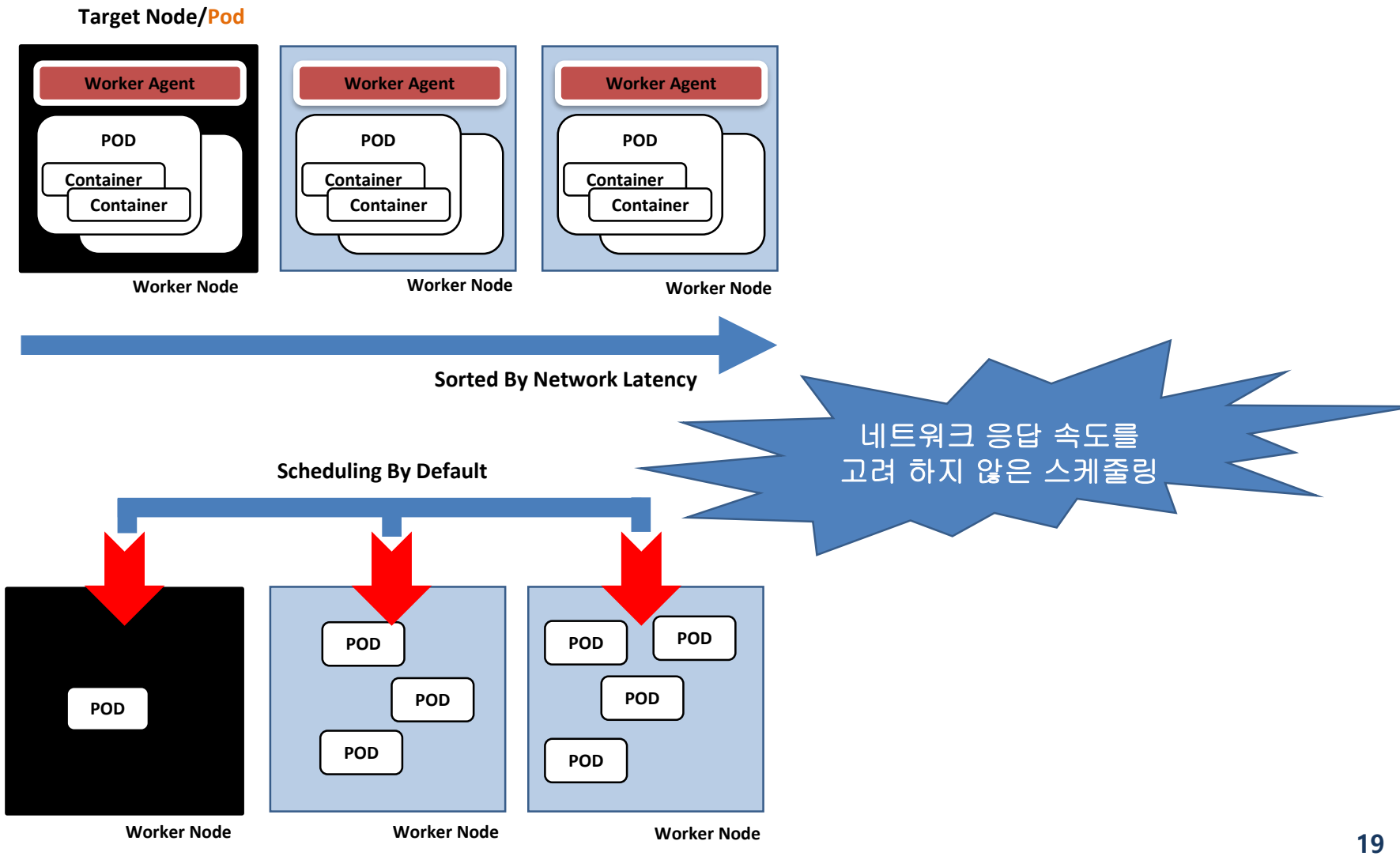
```
redis-config
maxmemory 30mb
maxmemory-policy allkeys-lru
bind 0.0.0.0
protected-mode no
```




```
redis_service.yaml
apiVersion: v1
kind: Service
metadata:
  name: redis-service
spec:
  type: NodePort
  selector:
    app: redis
  ports:
    - protocol: TCP
      port: 6379
      targetPort: 6379
```

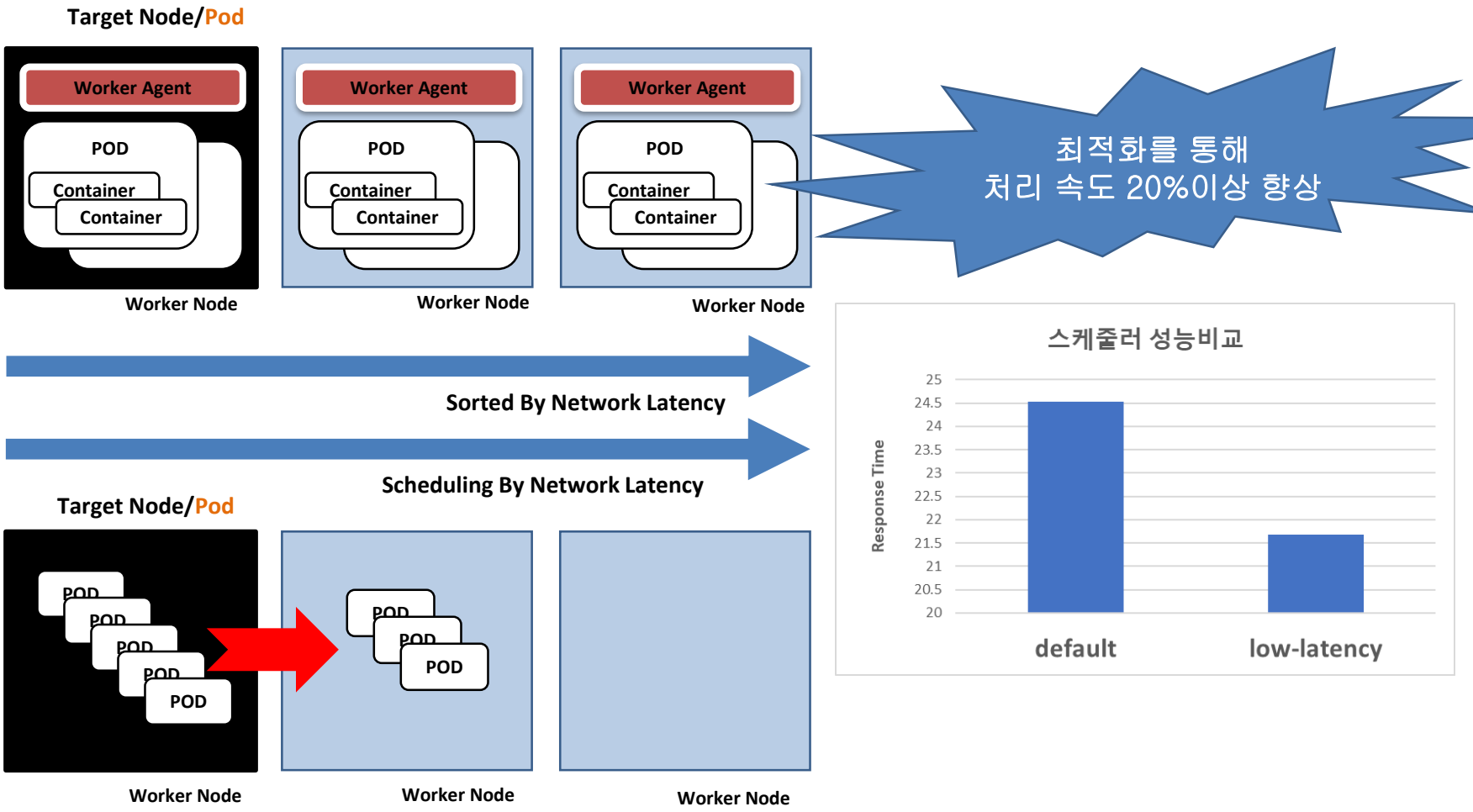
3 클라우드 엣지 컴퓨팅을 위한 전용 로컬 스케줄러 개발

GS-Scheduler : Low-Latency 처리 개념도

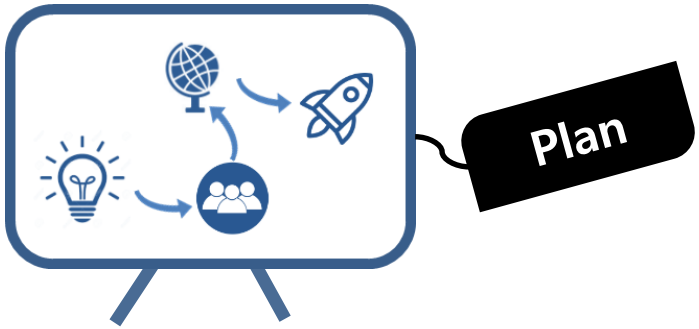


3 클라우드 엣지 컴퓨팅을 위한 전용 로컬 스케줄러 개발

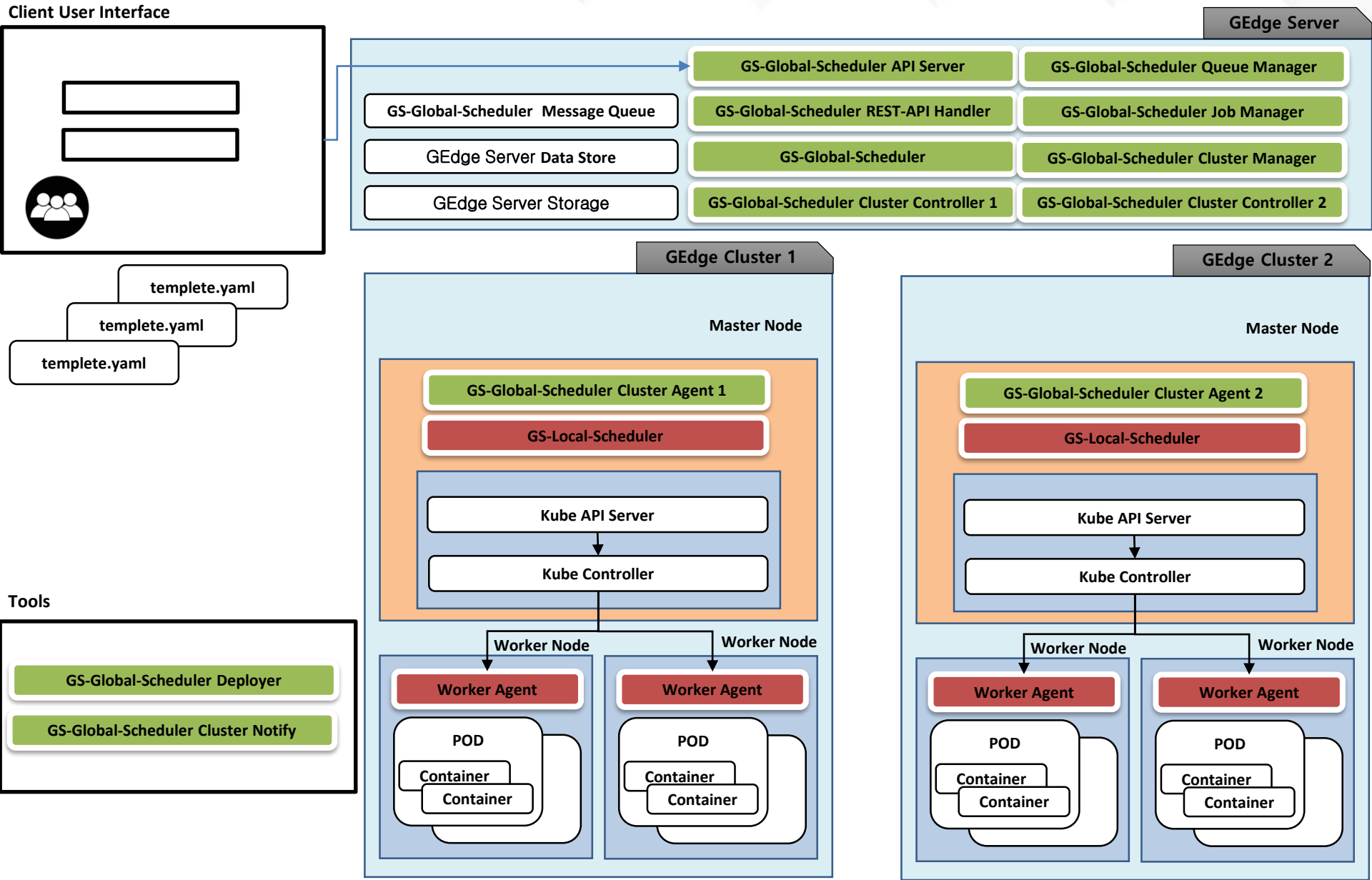
GS-Scheduler : Low-Latency 처리 개념도



클라우드 엣지 컴퓨팅을 위한 전용 글로벌 스케줄러 개발 계획



4 클라우드 엣지 컴퓨팅을 위한 전용 글로벌 스케줄러 개발 계획

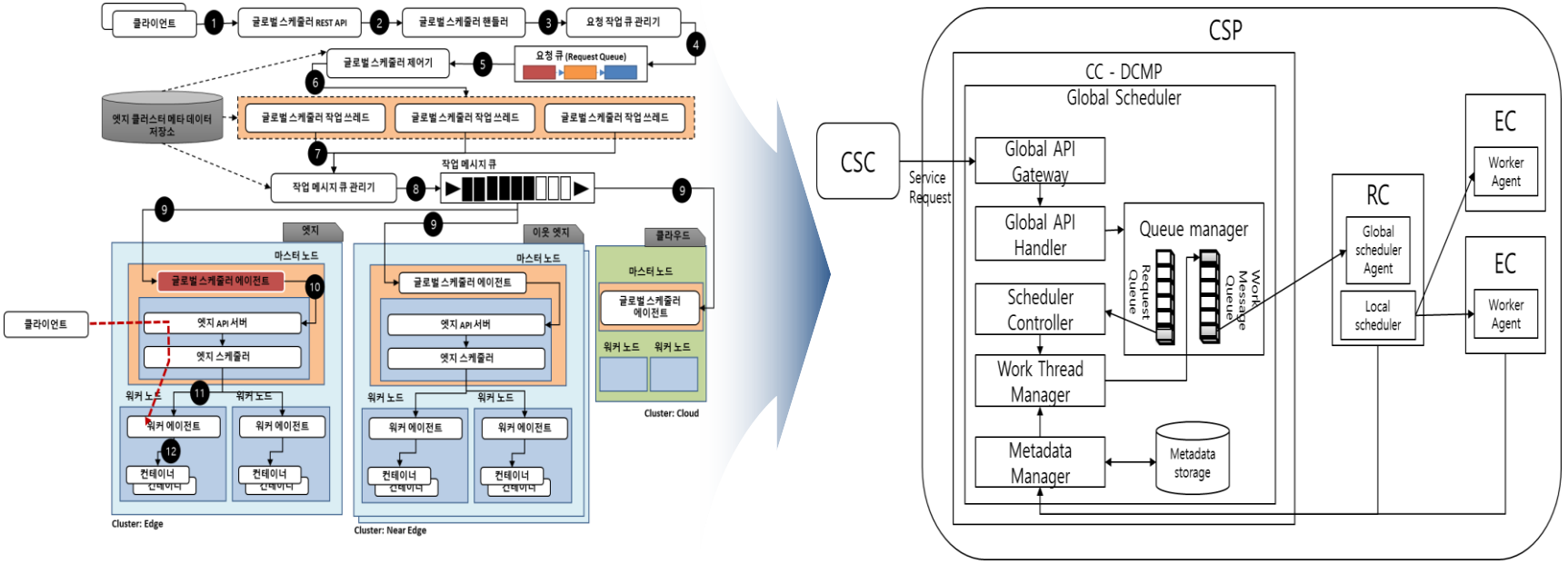


4 클라우드 엣지 컴퓨팅을 위한 전용 글로벌 스케줄러 개발 계획

국제표준특허 진행중

ITU-T SG13 WP2 Q19 – 기고 특허 반영(7월회의) 완료

The use case of global scheduling for distributed cloud

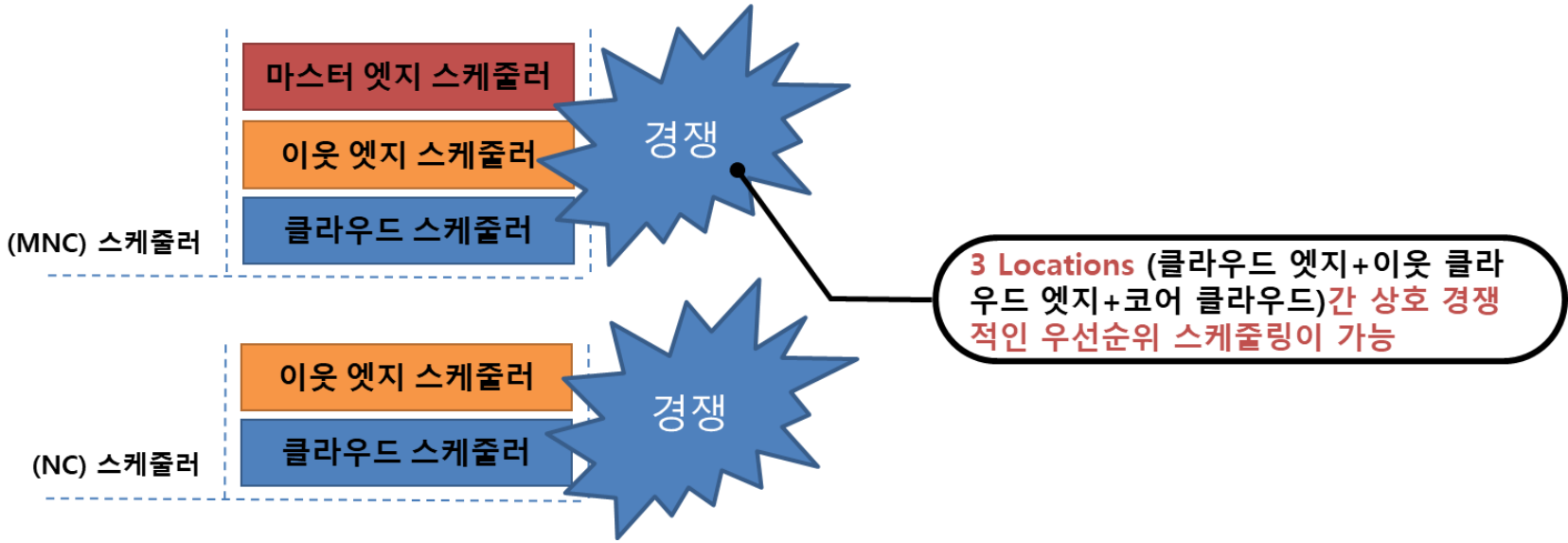


4 클라우드 엣지 컴퓨팅을 위한 전용 글로벌 스케줄러 개발 계획

엣지 글로벌 스케줄러 상세구조

Leveled 스케줄러

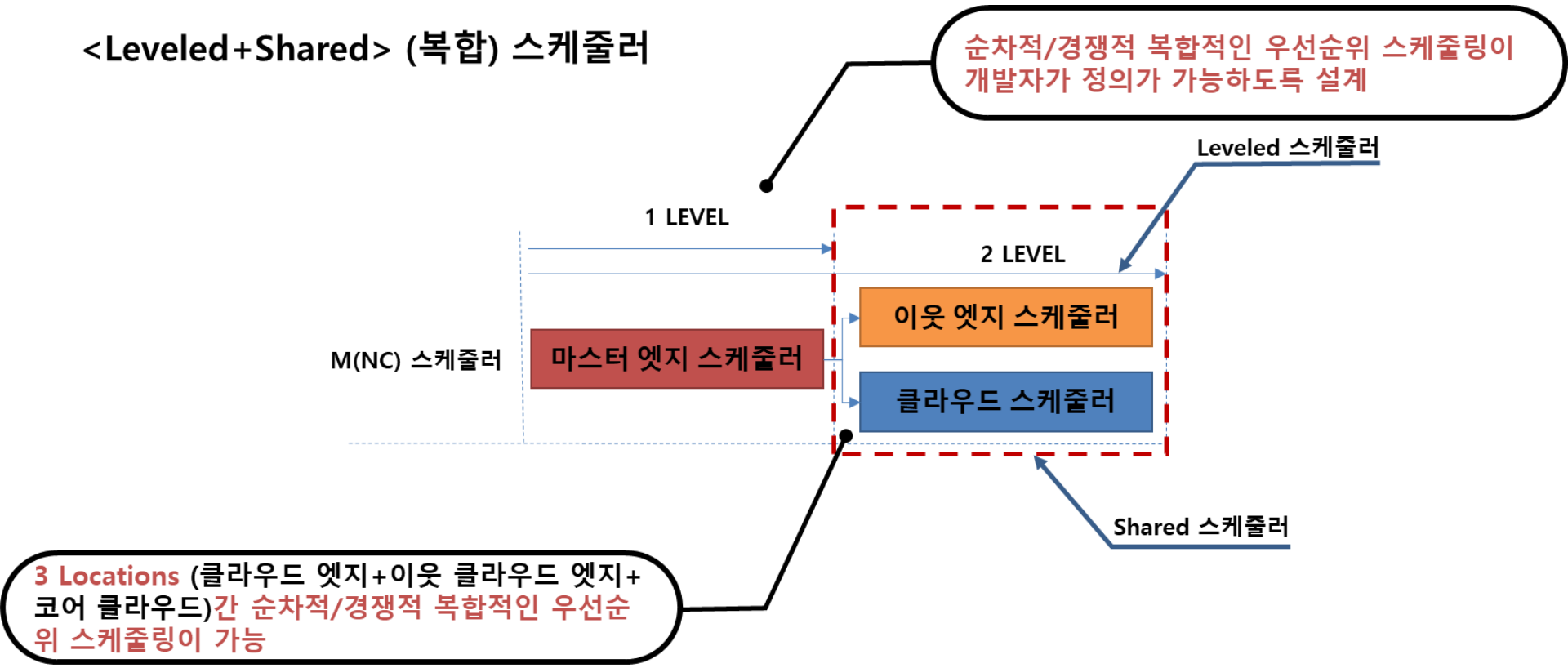
Shared 스케줄러



4 클라우드 엣지 컴퓨팅을 위한 전용 글로벌 스케줄러 개발 계획

GEdge Platform

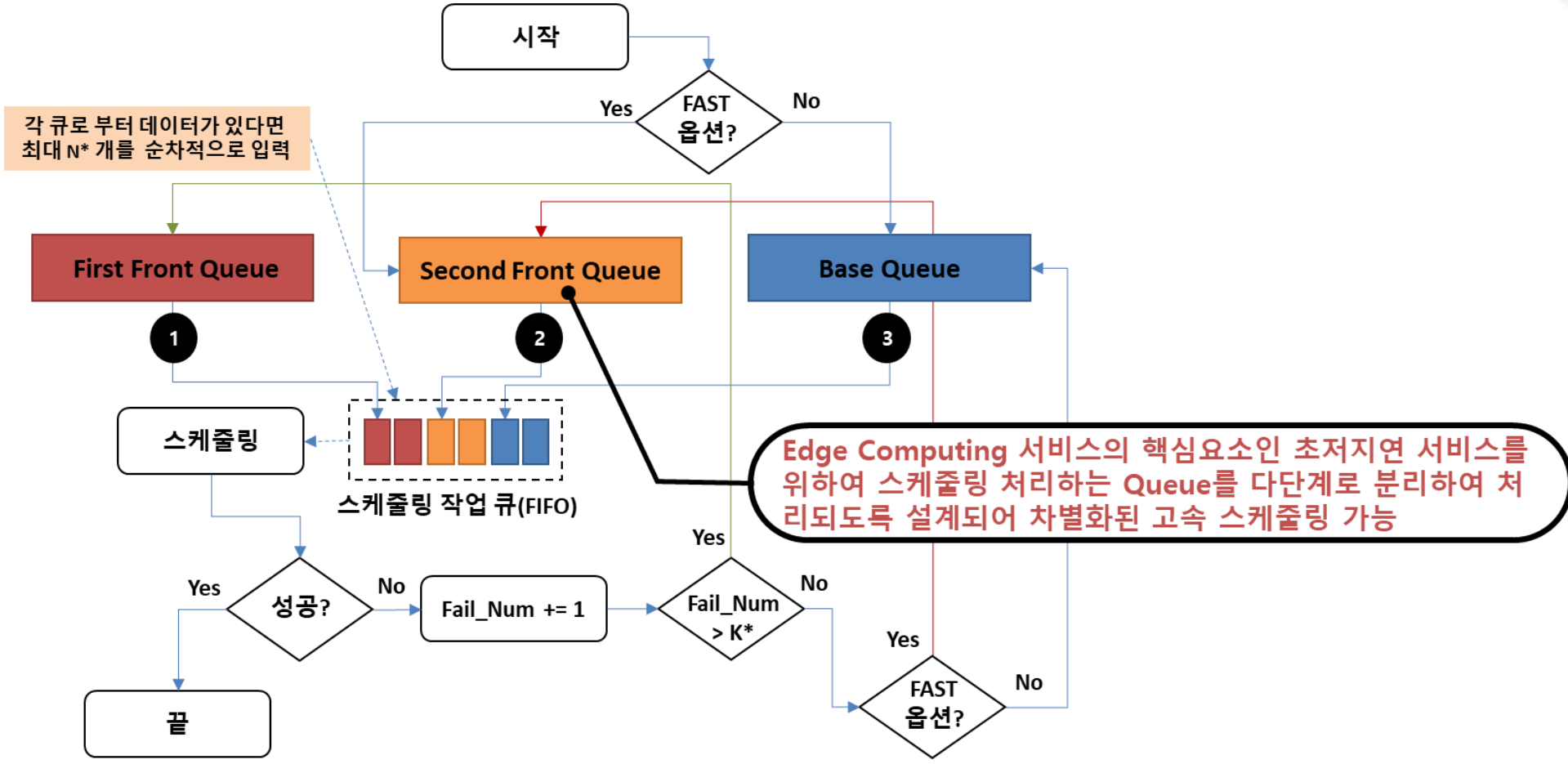
엣지 글로벌 스케줄러 상세구조



4 클라우드 엣지 컴퓨팅을 위한 전용 글로벌 스케줄러 개발 계획

GEdge Platform

엣지 글로벌 스케줄러 상세구조



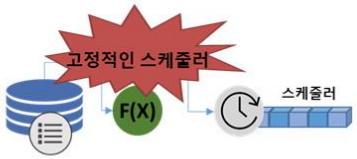
N*, K* Values are defined by System

4 클라우드 엣지 컴퓨팅을 위한 전용 글로벌 스케줄러 개발 계획

엣지 글로벌 스케줄러- 지능형 스케줄링 기능

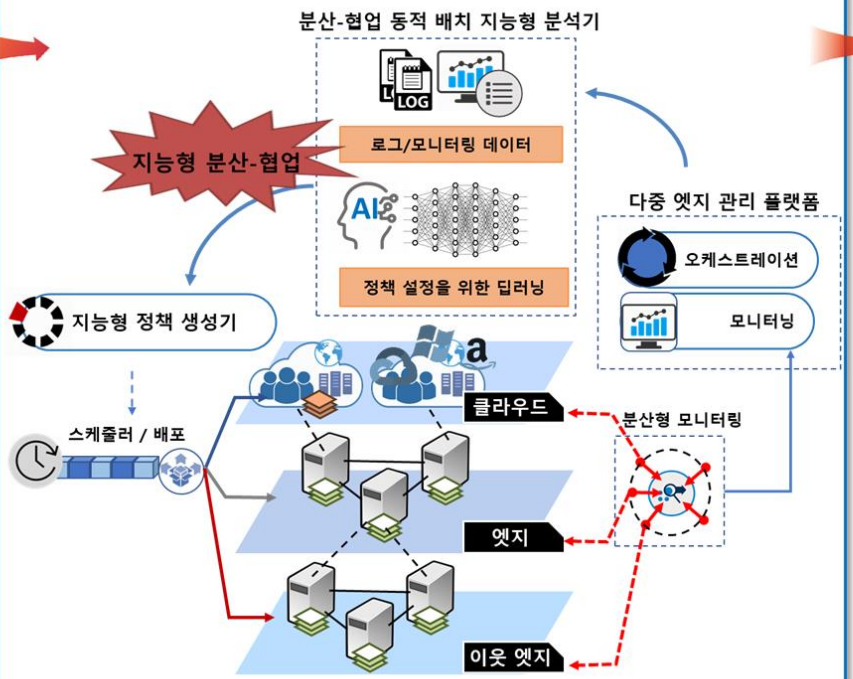
기존 기술 문제점

고정적 스케줄러 사용 → 응답 시간 증가



- 특정 로직에 따른 **고정적인 스케줄러**
- 서비스 환경 변화에 **비적응형**

핵심 개발 내용



개선 기술 특장점

지능형 분석기 기반 서비스 동적 배치 → 대기·지연 시간 최소화

- Near Cloud Edge/Core Cloud 포함한 **전역적인 분산처리** 가능
- 로그나 통계정보를 기반으로 **인공지능**을 이용한 **적응형 스케줄러**의 서비스 **동적 배치**의 최적화

감사합니다.

http://gedge-platform.github.io

GEdge Platform 코어 개발자
장수민(jsm@etri.re.kr)