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

최적 자원 배치를 위한 글로벌 스케줄링 기술

2022.12.20

GS-Scheduler 코어개발자

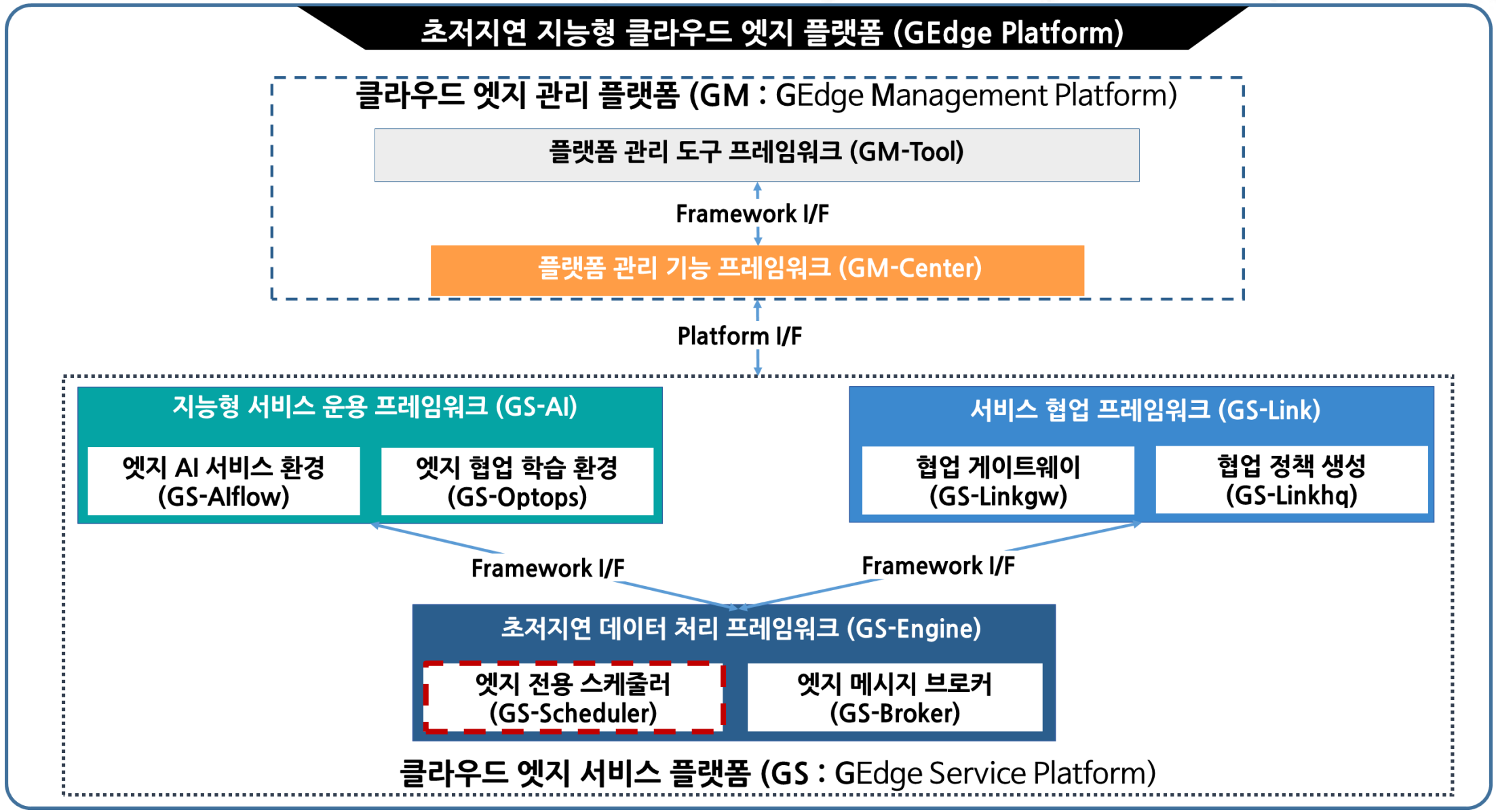
발표자 장수민(jsm@etri.re.kr)

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

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

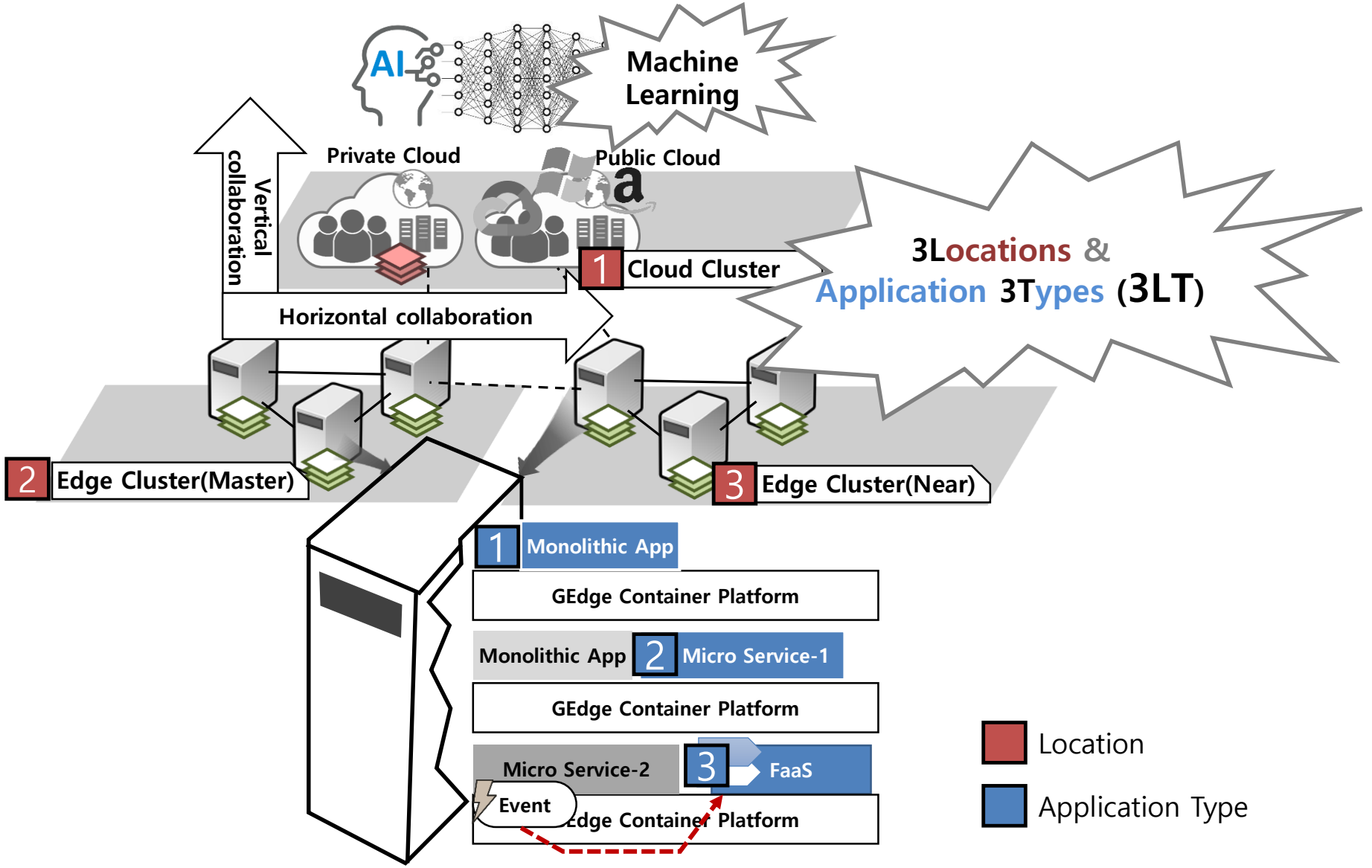
Contents

GEdge 플랫폼 내 GS-Scheduler의 포지셔닝

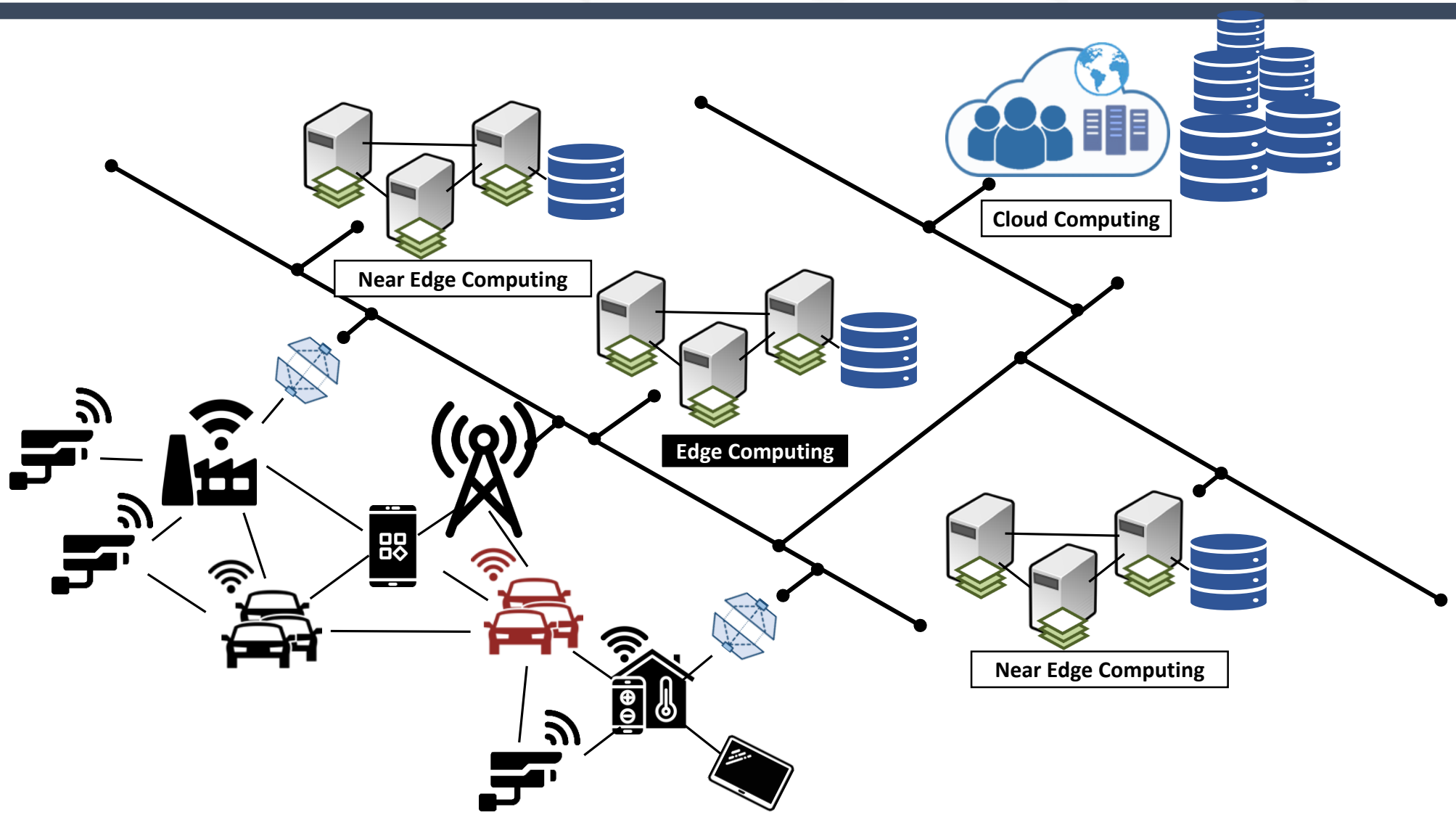


GS-Scheduler 개발 환경 및 개요

1 GS-Scheduler 개발 환경 및 개요



1 GS-Scheduler 개발 환경 및 개요

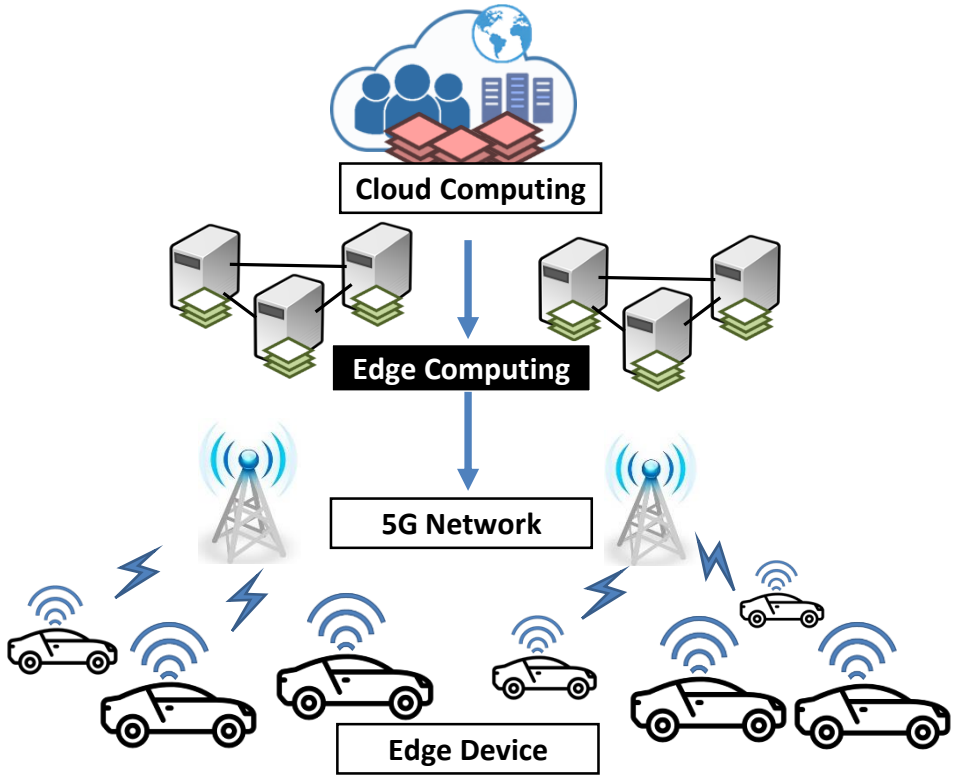


1 GS-Scheduler 개발 환경 및 개요

저지연 애플리케이션을 위한 엣지 컴퓨팅

저지연 애플리케이션 서비스 이슈

- 최근에 스마트 장치의 폭발적인 성장으로 **최종 사용자와 가능한 가깝게 데이터를 저장하고 처리하는 방식인 엣지 컴퓨팅**은 수요를 맞추기 힘들 정도로 빠르게 성장하는 분야
- 특히 **VR/AR, 자율 및 커넥티드 차량, 드론 제어, 원격 수술 및 의료 로봇 기기, 실시간 멀티플레이어 온라인 게임**과 같은 낮은 대기 시간 및 실시간 사용 사례의 경우, 낮은 대기 시간을 보장하고 백홀 대역폭 및 비용을 최소화 필요
- **엣지 컴퓨팅에서 가장 효과적으로 서비스가 가능**
- 저지연 어플리케이션은 클라우드 컴퓨팅보다 기지국 가장자리에 위치한 에지 컴퓨팅에서 서비스가 처리되고 클라이언트 디바이스와 5G를 통하여 통신이 이루어지는 것이 특징



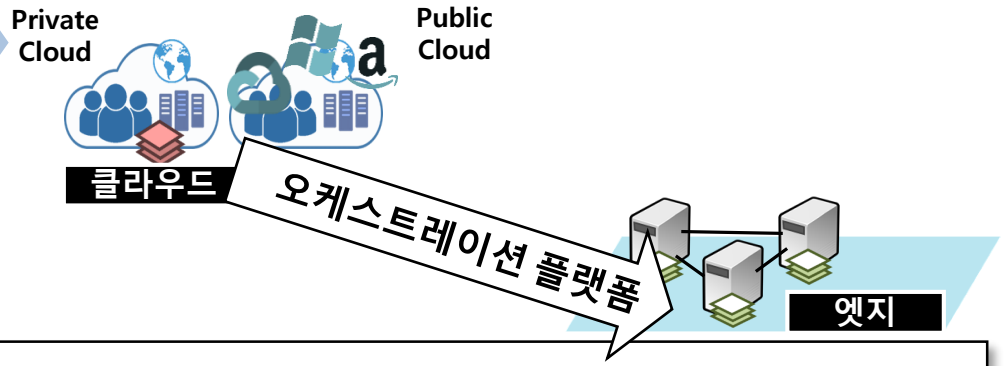
1 GS-Scheduler 개발 환경 및 개요

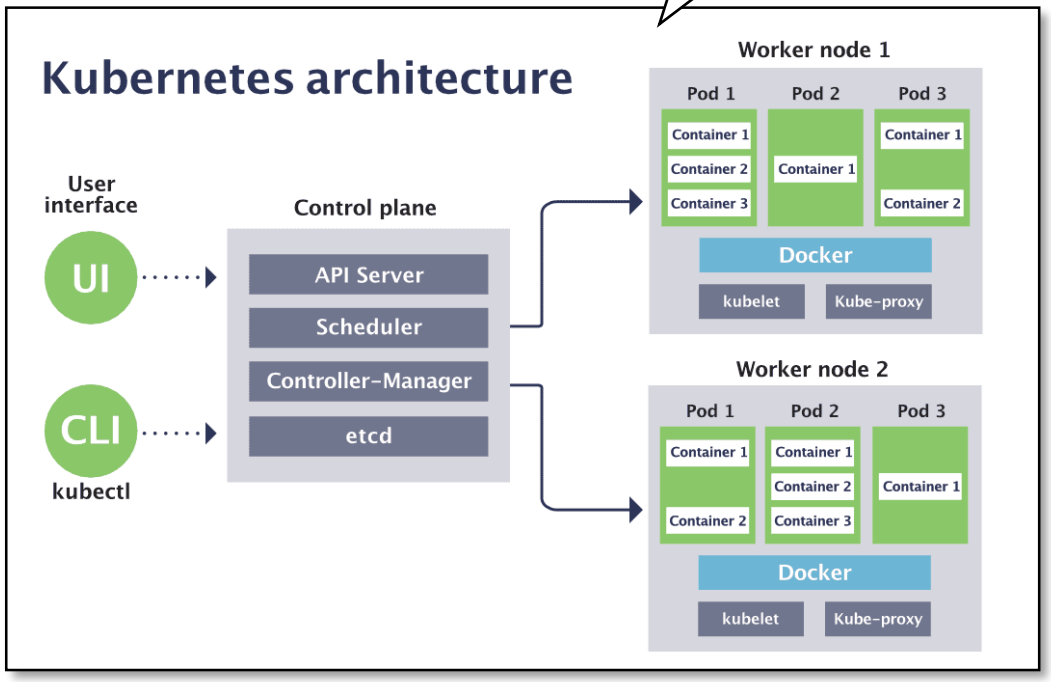
GEdge Platform

엣지 컴퓨팅 플랫폼

컨테이너 오케스트레이션 플랫폼

- 최근에 많은 저지연 애플리케이션 개발 및 운용자들은 엣지 컴퓨팅 처리를 위한 플랫폼으로 **컨테이너 오케스트레이션 플랫폼을 사용**
- 이러한 플랫폼은 **개별 컨테이너, 해당 기능 및 동적 환경의 모든 측면을 조정하고 구성하는 자동화된 프로세스들로 구성**되며 주된 역할은 **컨테이너 배포 및 확장, 네트워킹 및 유지 관리**.
- **컨테이너화**에는 소프트웨어 및 해당 종속성을 다른 구성 요소로부터 격리하는 방식으로 응용 프로그램을 패키징하는 작업이 포함
- 컨테이너는 **배포 및 유지 관리가 용이**하며 퍼블릭, 프라이빗 및 하이브리드 클라우드 환경에서 사용 가능
- 네트워크 엣지에서 실행되므로 **낮은 대기 시간, 글로벌 로드 밸런싱 및 낮은 대역폭 소비를 비롯한 여러 이점**을 제공
- 다양한 애플리케이션을 위하여 매우 많은 컨테이너를 수동으로 관리하는 것은 매우 어려움이 있어서 이를 해결하고자 사용



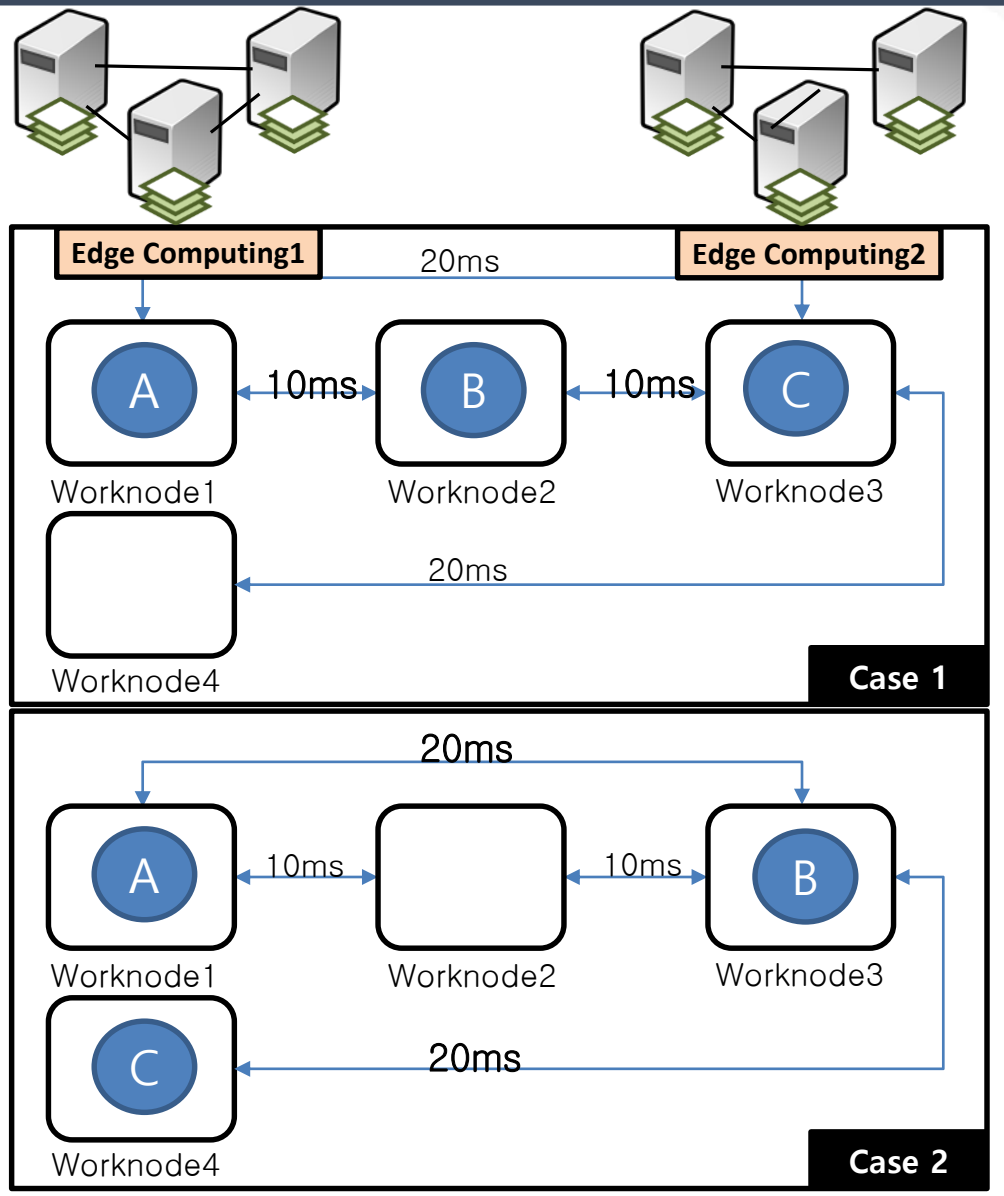


1 GS-Scheduler 개발 환경 및 개요

서비스 배포의 중요성

배포에 따른 서로 다른 처리속도

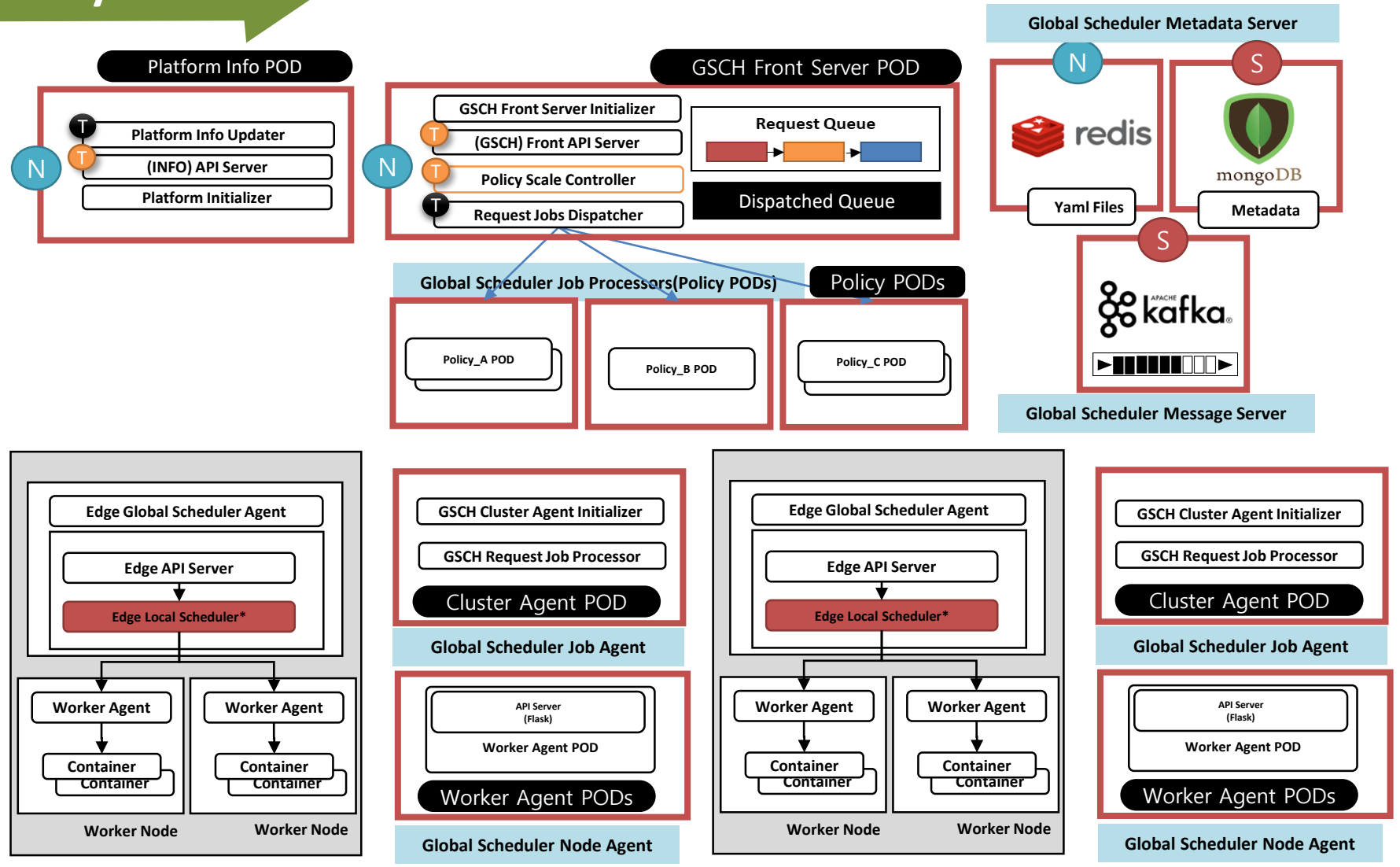
- 엔드투엔드 파이프라인의 형태로 배포될 때 서비스간 처리 속도를 고려하지 않으면 전체 어플리케이션의 성능에 문제가 생김
- **마이크로 서비스 A, B, C 순서적으로 처리되는 파이프라인 처리**를 할 경우 서로 다른 결과를 보여줌
- 예시는 각 마이크로 서비스 컨테이너 A, B, C가 특별한 자원이나 환경을 필요로 하여 서로 워커 노드에 할당되어야 하는 경우에 Case 2의 경우처럼 배포가 된다면 네트워크 속도가 Case 1에 비하여 네트워크를 통한 데이터 전송 측면만 고려해도 2배나 늦게 처리됨
- 저지연 애플리케이션의 경우 **서비스 QoS를 보장**하지 못하는 경우가 발생하여 서비스 제공에 실패할 가능성이 있는데 이러한 문제점을 완화하려면 적절한 배포 정책이 필요



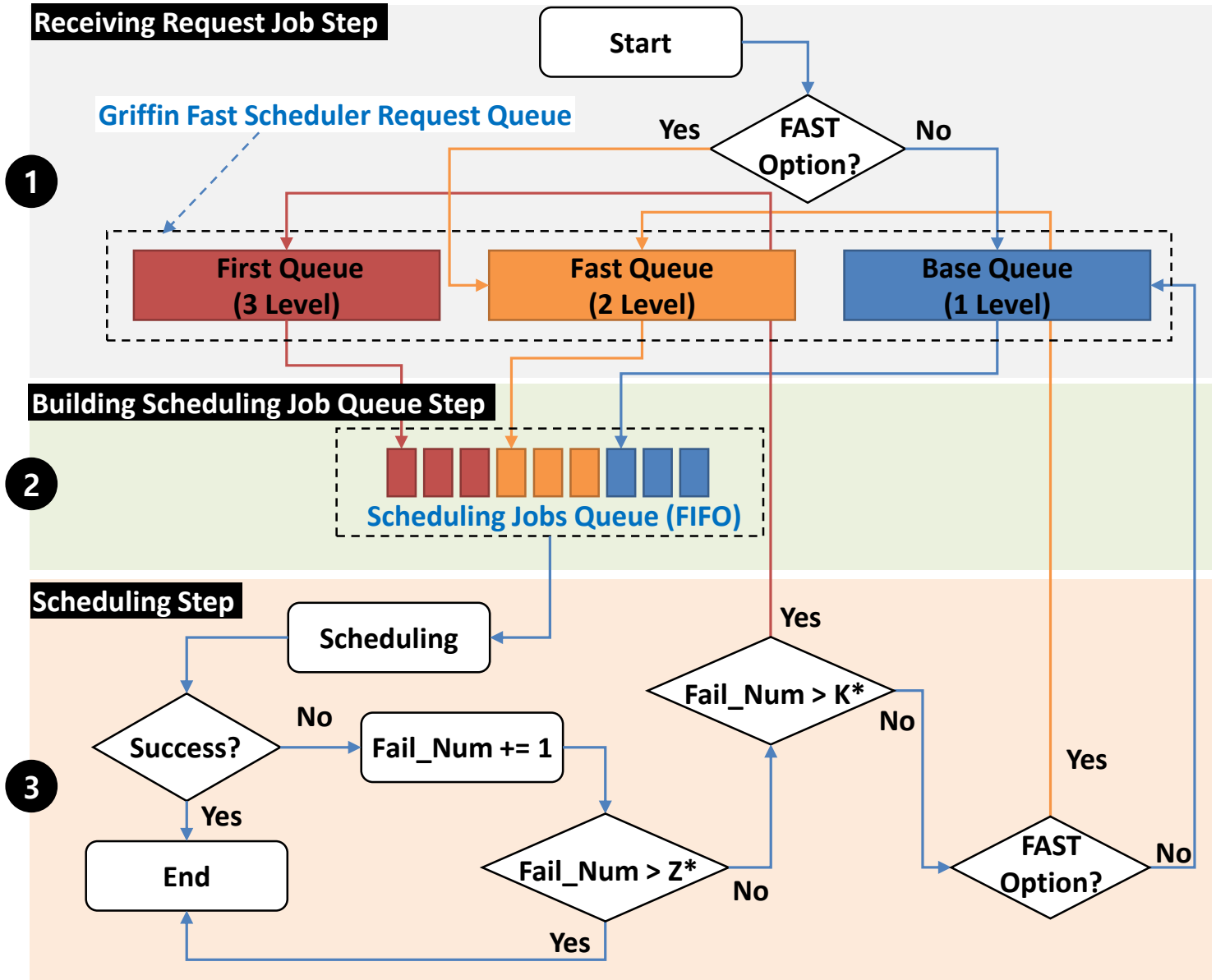
Ⅱ GS-Scheduler 핵심 기술

2 GS-Scheduler 핵심 기술

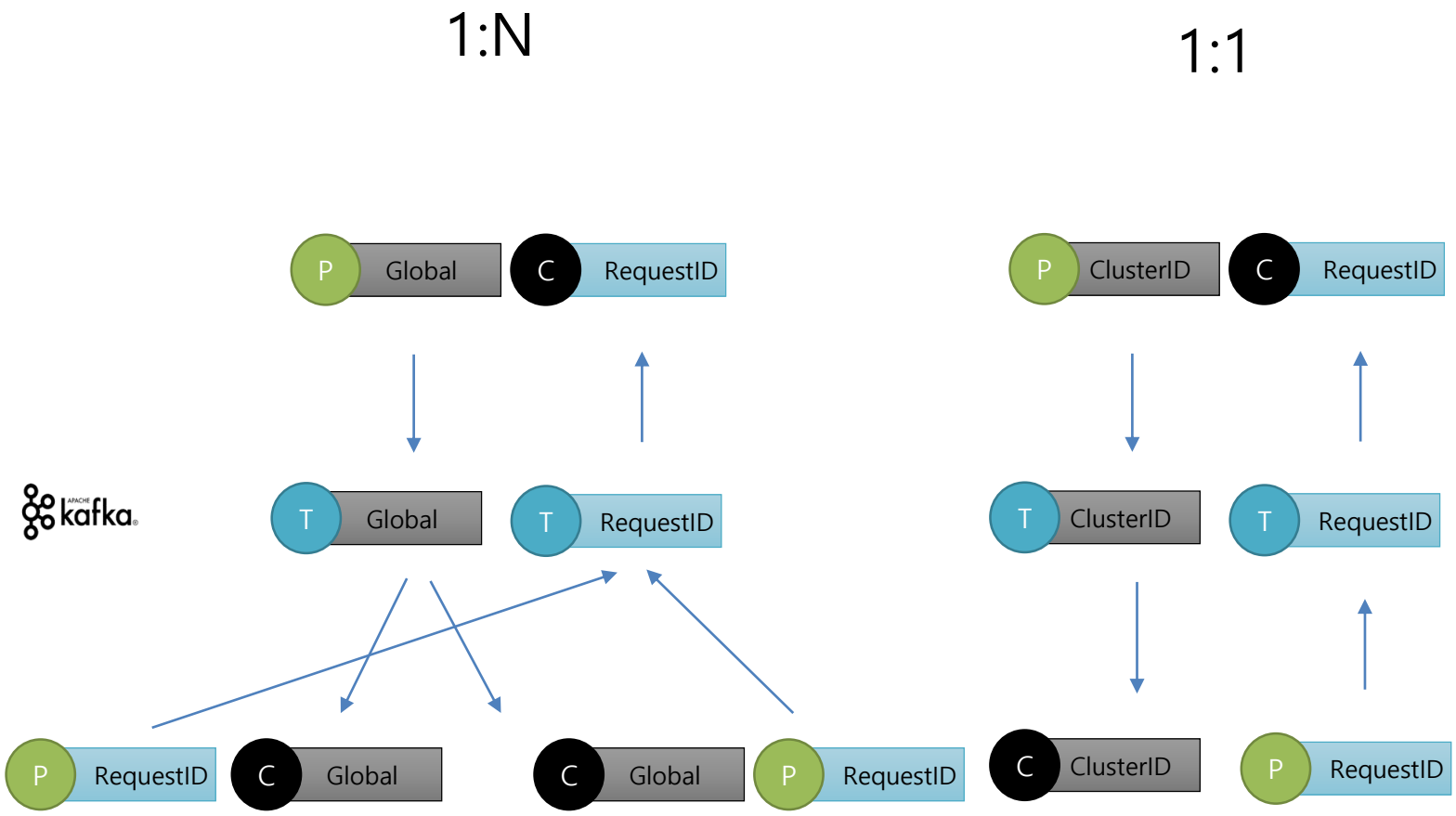
GEdge Scheduler System



2 GS-Scheduler 핵심 기술



2 GS-Scheduler 핵심 기술



2 GS-Scheduler 핵심 기술

GEdge Platform

클러스터간 메세지 전달 구조

Source	Type	cluster/node/pod
	Object	"c1" {"cluster":"c1","node":"n1"} {"cluster":"c1","pod":"p1"}
Target	Type	cluster(s)/node(s)/pod(s)
	Object	c1/[] [{"cluster":"c1","node(s)":n1/["n1"]}] [{"cluster":"c1","pod(s)":p1/["p1"]}]
HCode	0x0001	
LCode	0x0001	
Msg		

```
    '''-----------------------------------
        HCODE : 210  LCODE :1
    -----------------------------------'''
    def apply_GLowLatencyPriority_yaml(self, topicData):···

    '''-----------------------------------
        HCODE : 200  LCODE :1
    -----------------------------------'''
    def request_clusters_latency(self,topicData):···

    '''-----------------------------------
        HCODE : 300  LCODE :1
    -----------------------------------'''
    def request_clusters_available_resource(self,topicData):···

    '''-----------------------------------
        HCODE : 310  LCODE :1
    -----------------------------------'''
    def apply_GMostRequestedPriority_yaml(self, topicData):···

    '''-----------------------------------
        HCODE : 400  LCODE :1
    -----------------------------------'''
    def apply_GSelectedCluster_yaml(self, topicData):···

    '''-----------------------------------
        CLUSTER AGENT FUNCTIONS POINTER
    -----------------------------------'''
    functions={ 200:{ 1:request_clusters_latency },
                210:{ 1:apply_GLowLatencyPriority_yaml },
                300:{ 1:request_clusters_available_resource },
                310:{ 1:apply_GMostRequestedPriority_yaml },
                400:{ 1:apply_GSelectedCluster_yaml }
              }
```

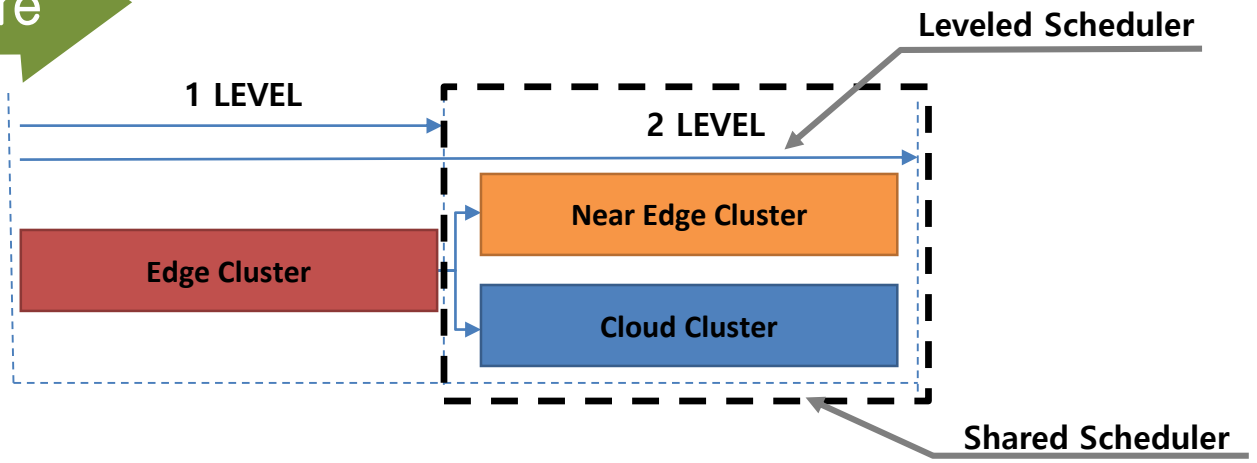
2 GS-Scheduler 핵심 기술

멀티 클러스터 엣지 스케줄러

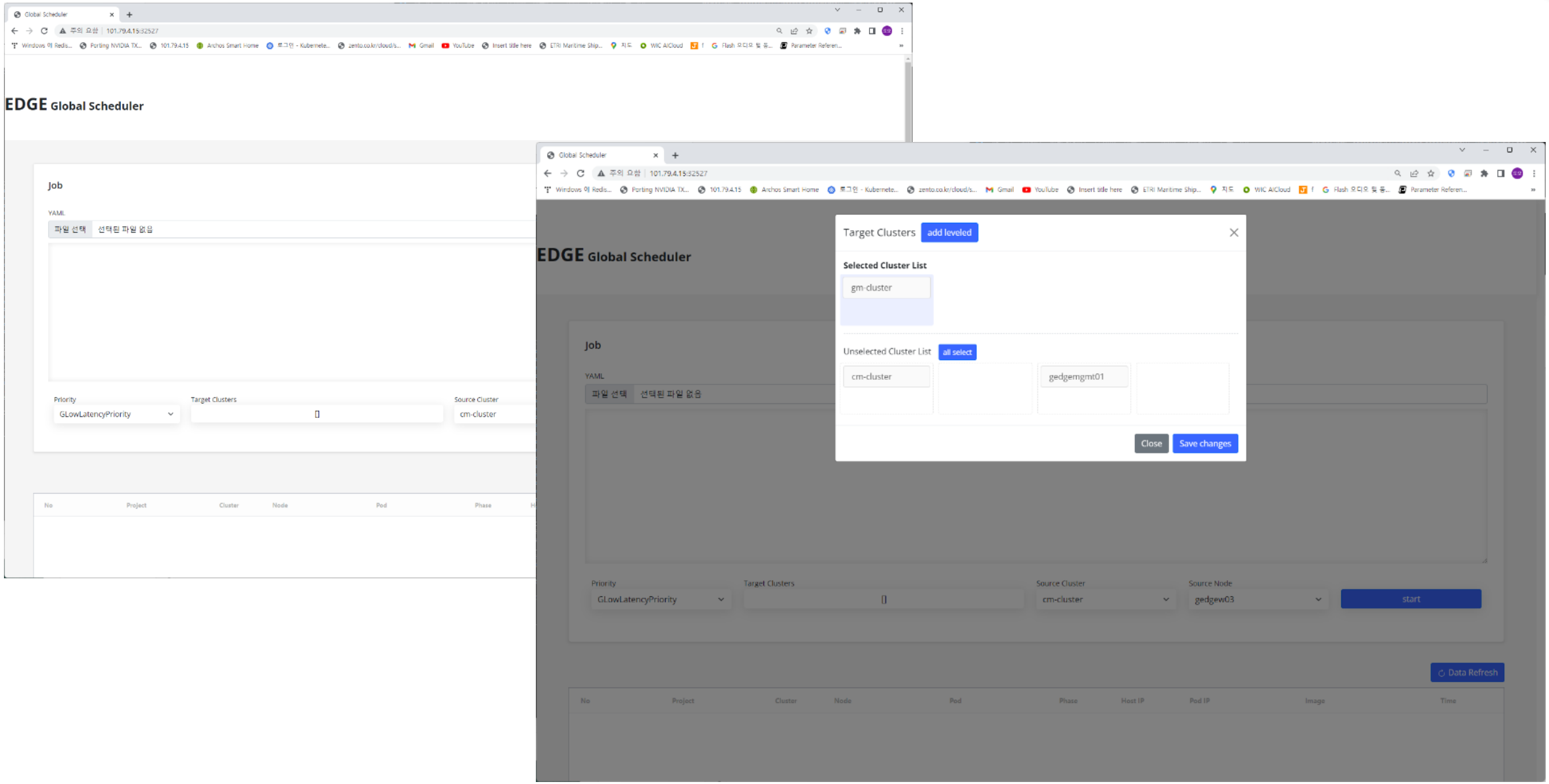
복합 (Leveled/Shared) 스케줄러

- 개발된 엣지 스케줄러는 **멀티 클러스터들을 순차적으로 배포 처리** 가능하도록 하는 **Leveled 스케줄러** 형식과 **두개 이상의 클러스터들을 경쟁시켜서 최적의 노드를 선택** 가능한 **Shared 스케줄러** 형식이 복합된 구조로 설계 및 개발
- 예시는 1Level부터 2Level로 순차적으로 처리하면서 2level에서는 이웃 엣지 클러스터와 클라우드들 간에 정책의 우선순위경쟁을 통한 프로세스 배포가 요청된 정책의 스케줄러 구성된 예시

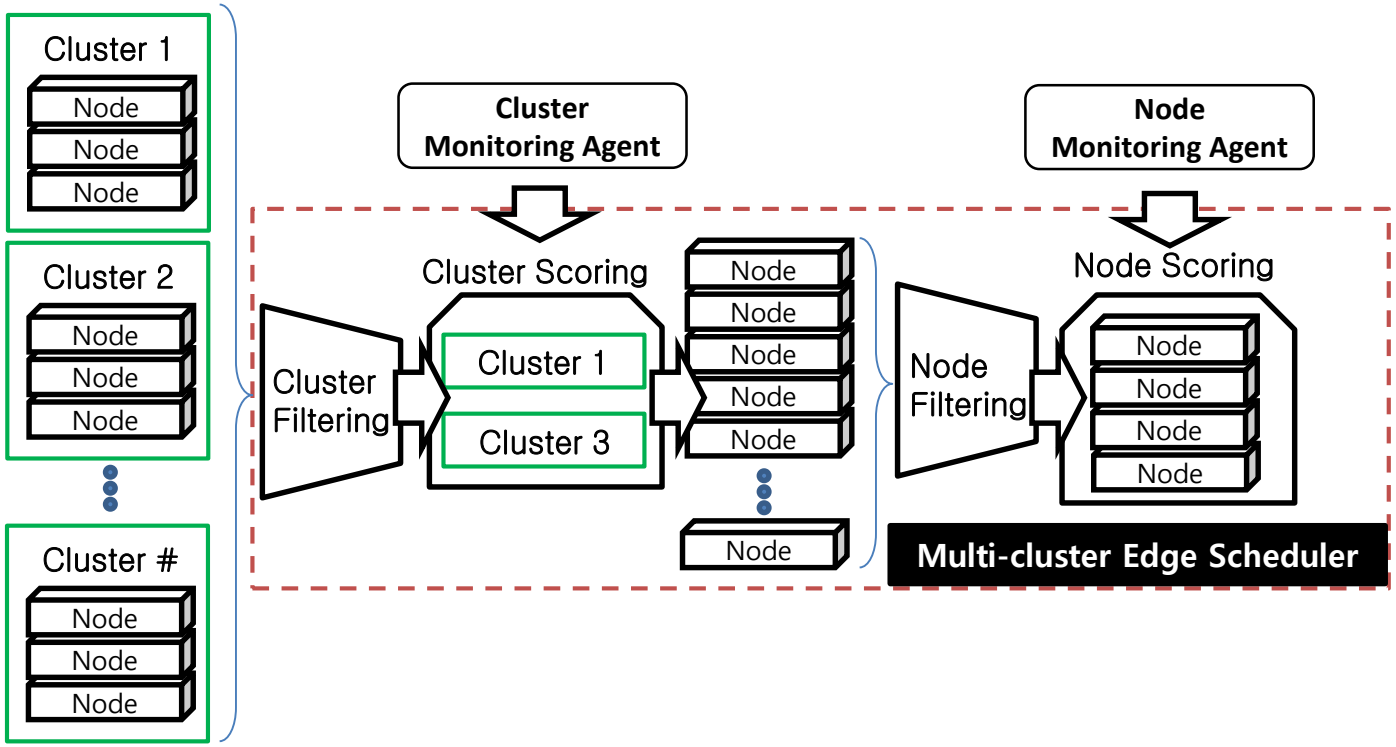
GEdge Scheduler Structure



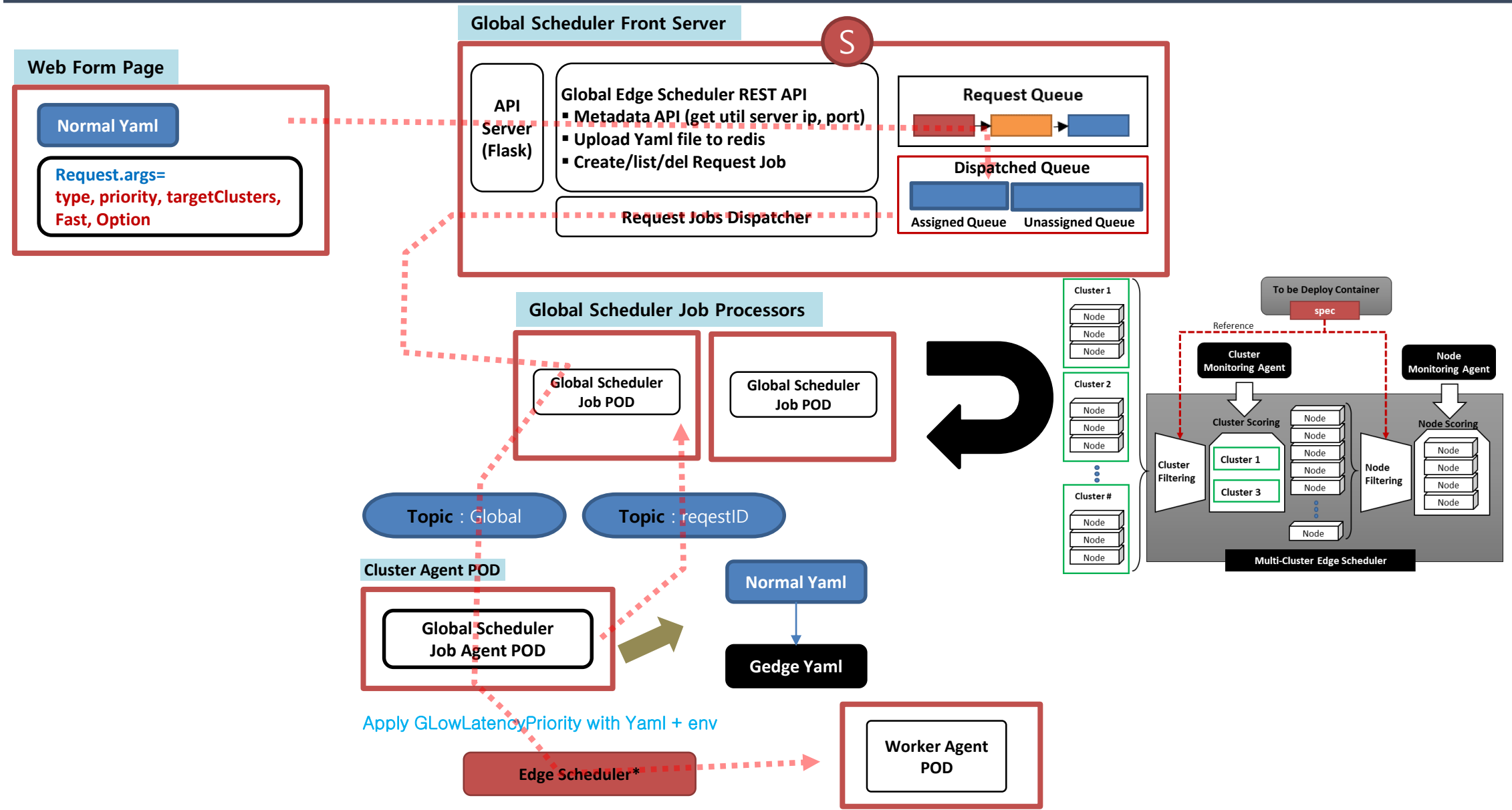
2 GS-Scheduler 핵심 기술



2 GS-Scheduler 핵심 기술



2 GS-Scheduler 핵심 기술



2 GS-Scheduler 핵심 기술

GEdge Platform

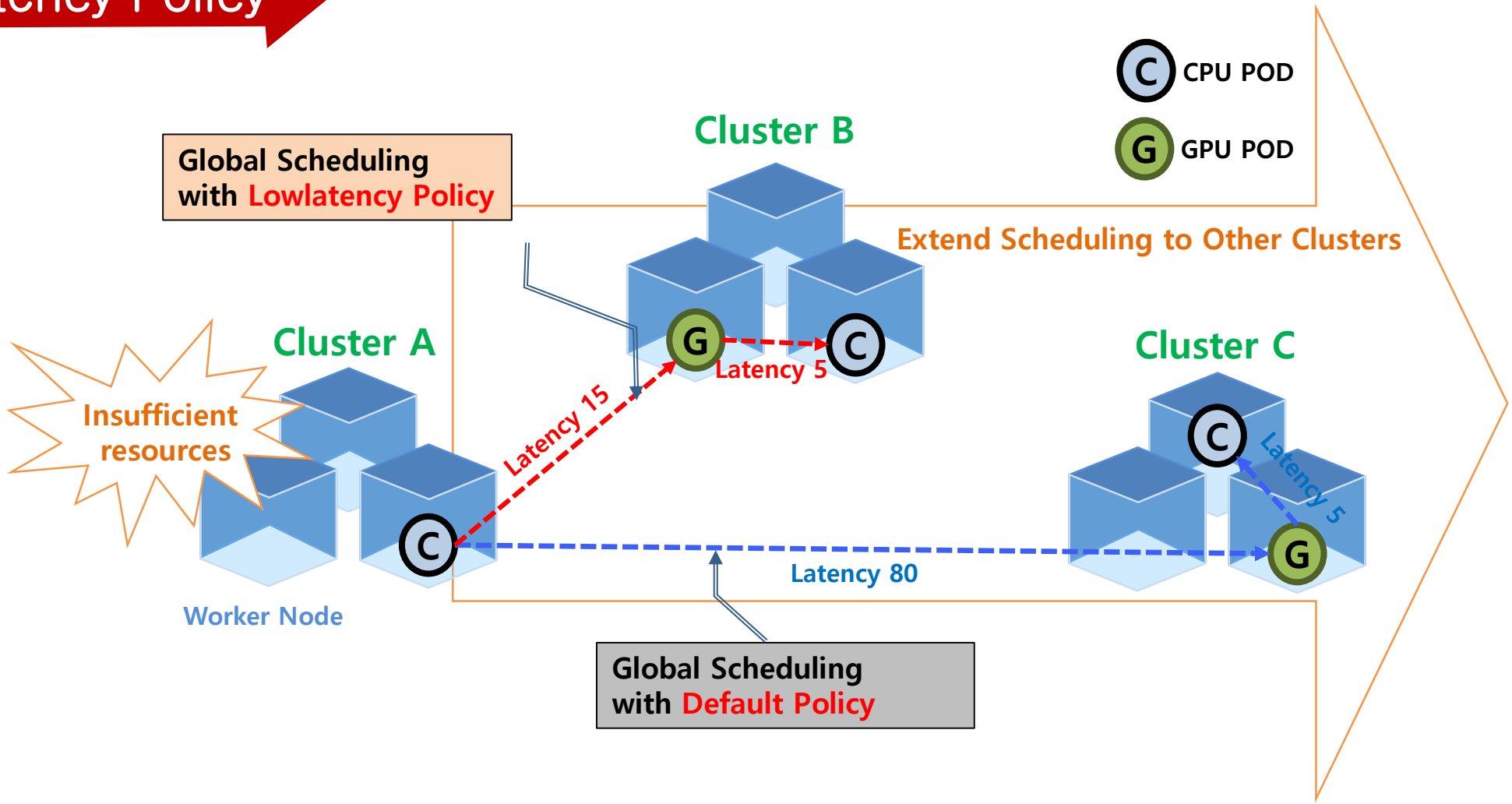
```
@app.route(f'{PREFIX}/page/low',methods=['GET','POST'])
def create_html_request_queue_insert_page():...

@app.route(f'{PREFIX}/page/most',methods=['GET','POST'])
def create_html_request_queue_insert_page2():...

@app.route(f'{PREFIX}/page/select',methods=['GET','POST'])
def create_html_request_queue_insert_page3():...

@app.route(f'{PREFIX}/test/low', methods=['POST'])
def create_scheduling_job():...

@app.route(f'{PREFIX}/test/most', methods=['POST'])
def create_scheduling_job2():...


@app.route(f'{PREFIX}/test/select', methods=['POST'])
def create_scheduling_job3():...


'''-----------------------------------------------------------------------------------------------------------
        DISPATCH REQUEST
-----------------------------------------------------------------------------------------------------------'''

@app.route(f'{PREFIX}/dispatched-queue/policies/<policy>', methods=['GET'])
def schedule_dispatched_request(policy):...

'''-----------------------------------------------------------------------------------------------------------
        REQUEST
-----------------------------------------------------------------------------------------------------------'''

@app.route(f'{PREFIX}/dispatched-queue/request_jobs/<request_id>/status/<changed_status>', methods=['PUT'])
def update_dispatched_queue_status(request_id,changed_status):...

'''-----------------------------------------------------------------------------------------------------------
    SCALE POLICY POD
-----------------------------------------------------------------------------------------------------------'''
@app.route(f'{PREFIX}/policys/<policy>/replicas/<replicas>', methods=['PUT'])
def update_policy_scale(policy,replicas):...
```

Processing Requests Job

Dispatch Requests Job & Update Status

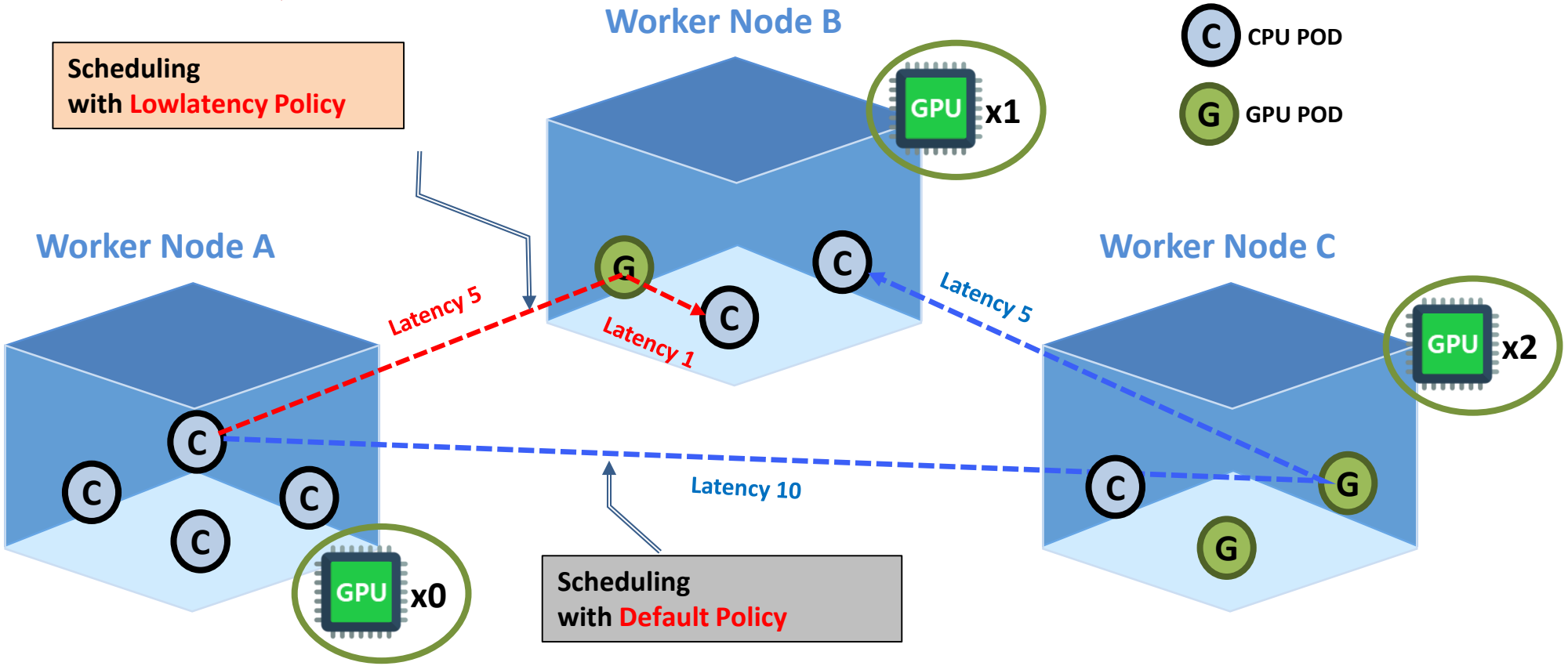
2 GS-Scheduler 핵심 기술

Lowlatency Policy



2 GS-Scheduler 핵심 기술

Lowlatency Policy



2 GS-Scheduler 핵심 기술

GEdge Platform

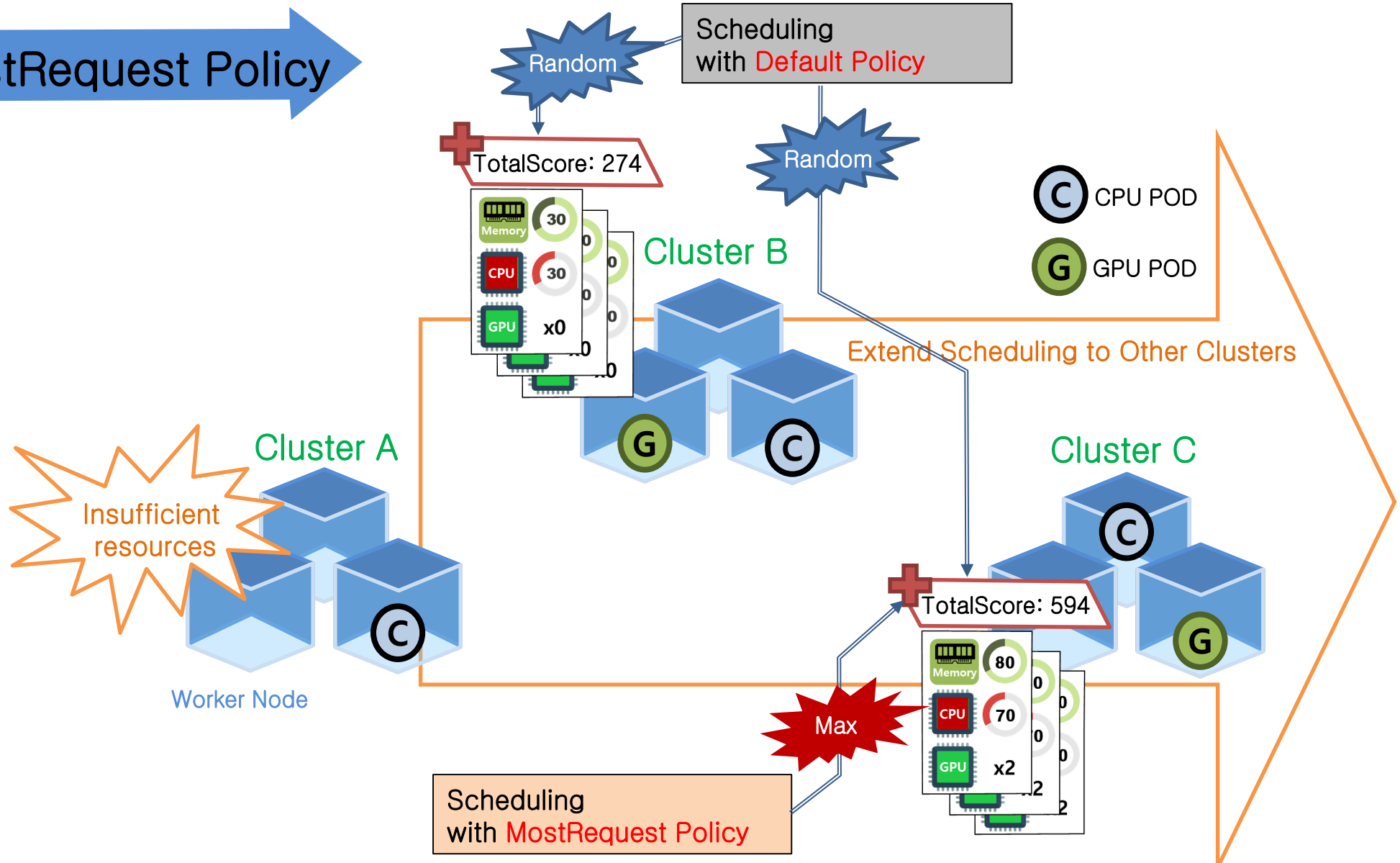
```
'''
{'requestID': 'req-f6720a0e-e3df-455a-825d-f8c80cedc2d9',
 'date': '2021-10-18 13:46:30', 'status': 'create',
 'fileID': 'b469e54a-721f-4c55-b43e-d09088556031', 'failCnt': 0,
 'env': {
        'type': 'global',
        'targetClusters': ['c1', ['c2', 'c3'], 'c4'],
        'priority': 'GLowLatencyPriority',
        'option': {
            'sourceCluster': 'c1',
            'sourceNode': 'a-worker-node01'
        }
       }
}
'''
```

```
class GLowLatencyPriority_Job:
    def __init__(self,request_data_dic):
        self.job_name       = lowDefine.SELF_POLICY_NAME
        self.requestDataDic = request_data_dic
        self.requestID      = request_data_dic['requestID']
        self.fileID         = request_data_dic['fileID']
        self.failCnt        = request_data_dic['failCnt']

        self.env            = request_data_dic['env']
        self.targetClusters = self.env['targetClusters']
        self.sourceCluster  = self.env['option']['sourceCluster']
        self.sourceNode     = self.env['option']['sourceNode']
        self.producer       = KafkaProducer(acks=0,compression_type='gzip',
                              bootstrap_servers=[gDefine.KAFKA_SERVER_URL],
                              value_serializer=lambda x: dumps(x).encode('utf-8'))

    def check_for_fault_response_msg(self, res): ···

    def send_clusters_latency_request_msg_to_cluster_agents(self,clusters): ···

    def wait_clusters_latency_response_msg_from_cluster_agents(self): ···

    def send_apply_yaml_request_msg_to_cluster_agent(self,cluster): ···

    def wait_response_msg_of_apply_yaml_request_from_cluster_agents(self): ···

    def wait_response_msg_from_request_id_topic(self): ···

def read_dispatched_queue(): ···

def request_job_processor(): ···
```

2 GS-Scheduler 핵심 기술

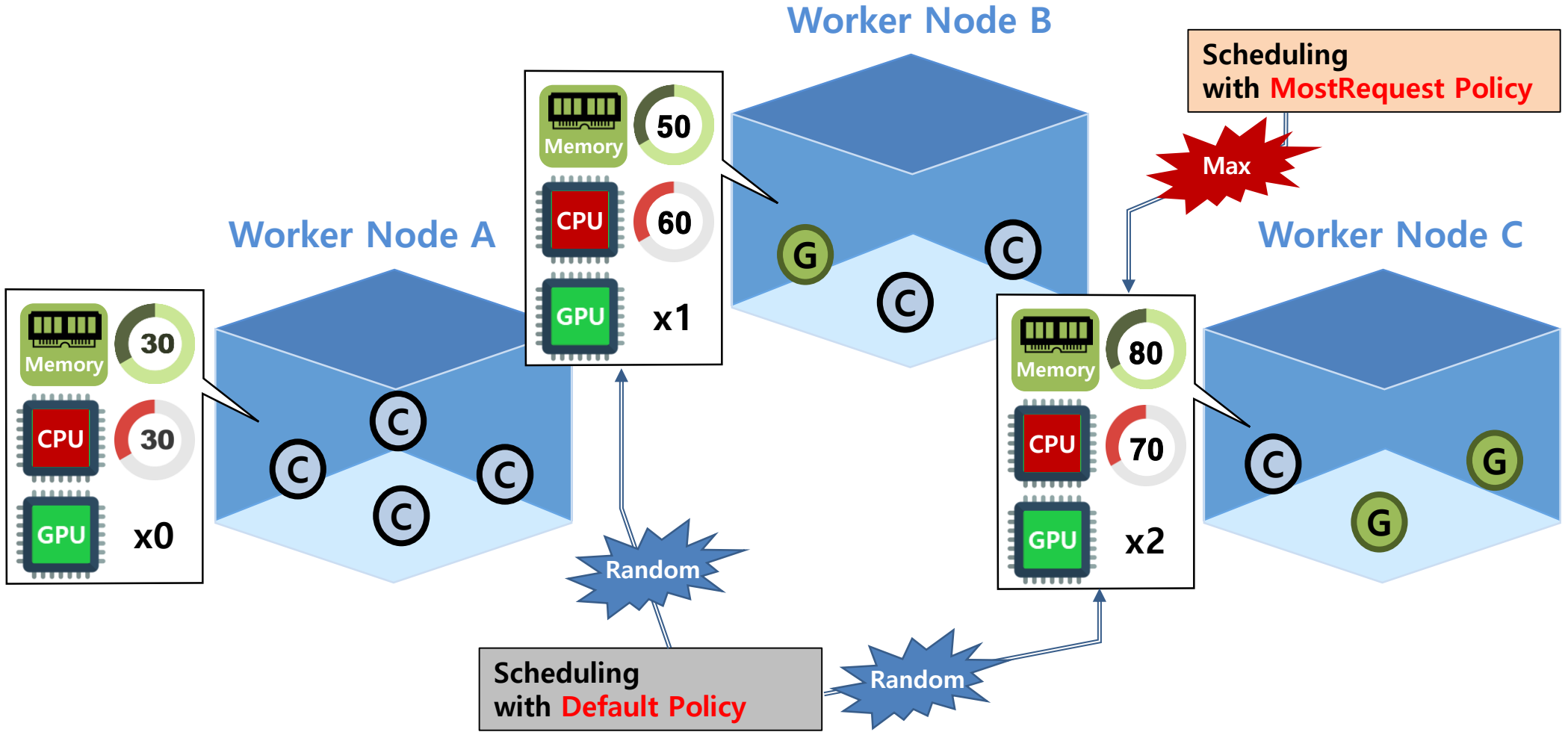
GEdge Platform

MostRequest Policy



2 GS-Scheduler 핵심 기술

MostRequest Policy



2 GS-Scheduler 핵심 기술

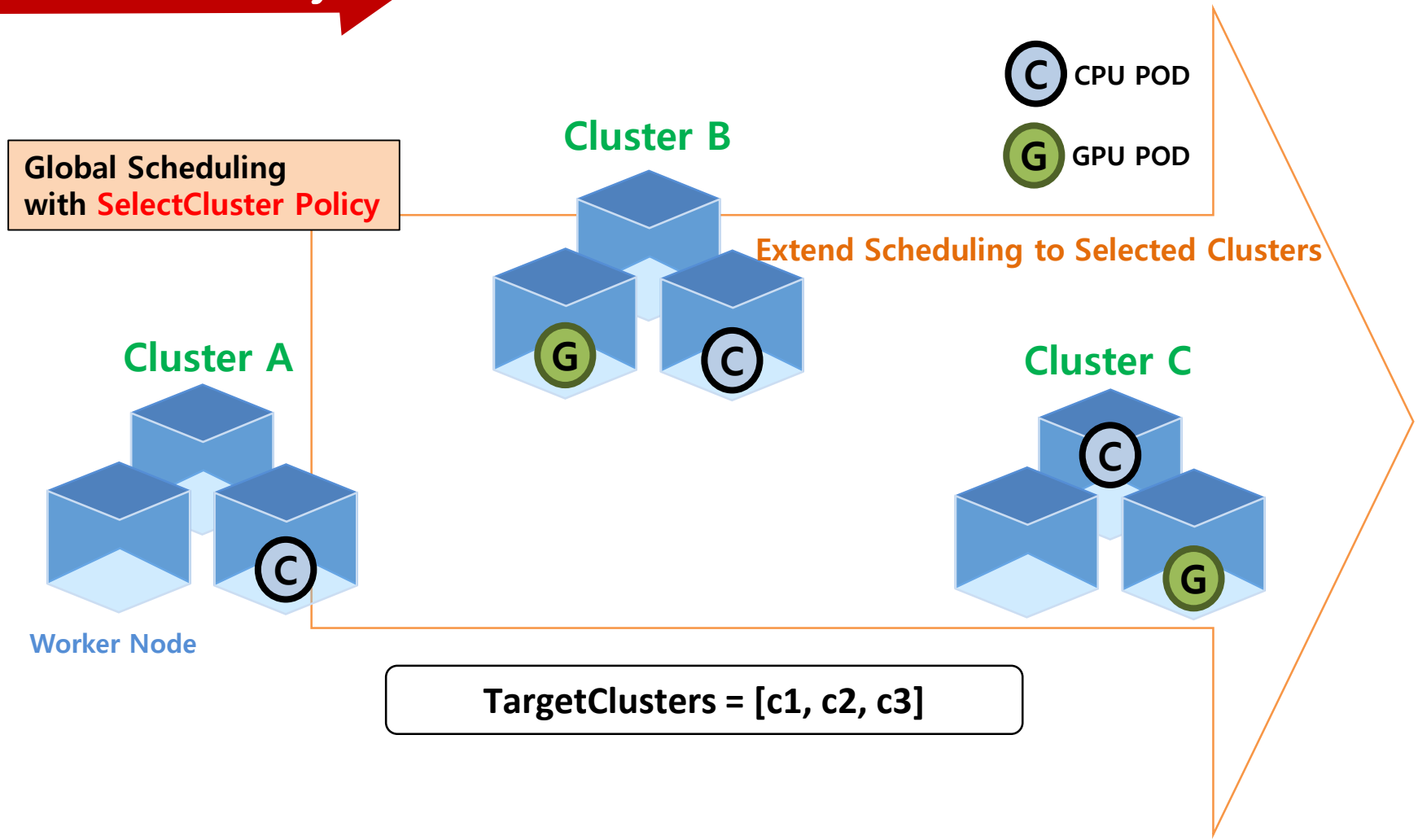
GEdge Platform

```
'''
{'requestID': 'req-f6720a0e-e3df-455a-825d-f8c80cedc2d9',
 'date': '2021-10-18 13:46:30', 'status': 'create',
 'fileID': 'b469e54a-721f-4c55-b43e-d09088556031', 'failCnt': 0,
 'env': {
         'type': 'global',
         'targetClusters': ['c1', ['c2', 'c3'], 'c4'],
         'priority': 'GMostRequestedPriority',
        }
}
'''
```

```
class GMostRequestedPriority_Job:
    def __init__(self,request_data_dic):
        self.job_name = mostDefine.SELF_POLICY_NAME
        self.requestDataDic = request_data_dic
        self.requestID=request_data_dic['requestID']
        self.fileID=request_data_dic['fileID']
        self.failCnt=request_data_dic['failCnt']

        self.env=request_data_dic['env']
        self.targetClusters=self.env['targetClusters']

        self.producer= KafkaProducer(acks=0,compression_type='gzip',
                        bootstrap_servers=[gDefine.KAFKA_SERVER_URL],
                        value_serializer=lambda x: dumps(x).encode('utf-8'))

        self.redis = redisController(gDefine.REDIS_ENDPOINT_IP,gDefine.REDIS_ENDPOINT_PORT)
        self.redis_conn = self.redis.redisConn
        self.yamlfile = self.get_yaml_file_from_redis(self.fileID)
        self.yaml_dic_list = yaml.load_all(self.yamlfile,Loader=yaml.FullLoader)

        self.GPUFilter = self.is_necessary_GPU_filter()

    def get_yaml_file_from_redis(self,yaml_key):···

    def is_necessary_GPU_filter(self) :···

    def check_for_fault_response_msg(self, res):···

    def send_clusters_available_resource_from_cluster_agents(self,clusters):···

    def wait_clusters_available_resource_from_cluster_agents(self,clusters):···

    def send_apply_yaml_request_msg_to_cluster_agent(self,cluster):···

    def wait_response_msg_of_apply_yaml_request_from_cluster_agents(self):···

    def wait_response_msg_from_request_id_topic(self):···
```

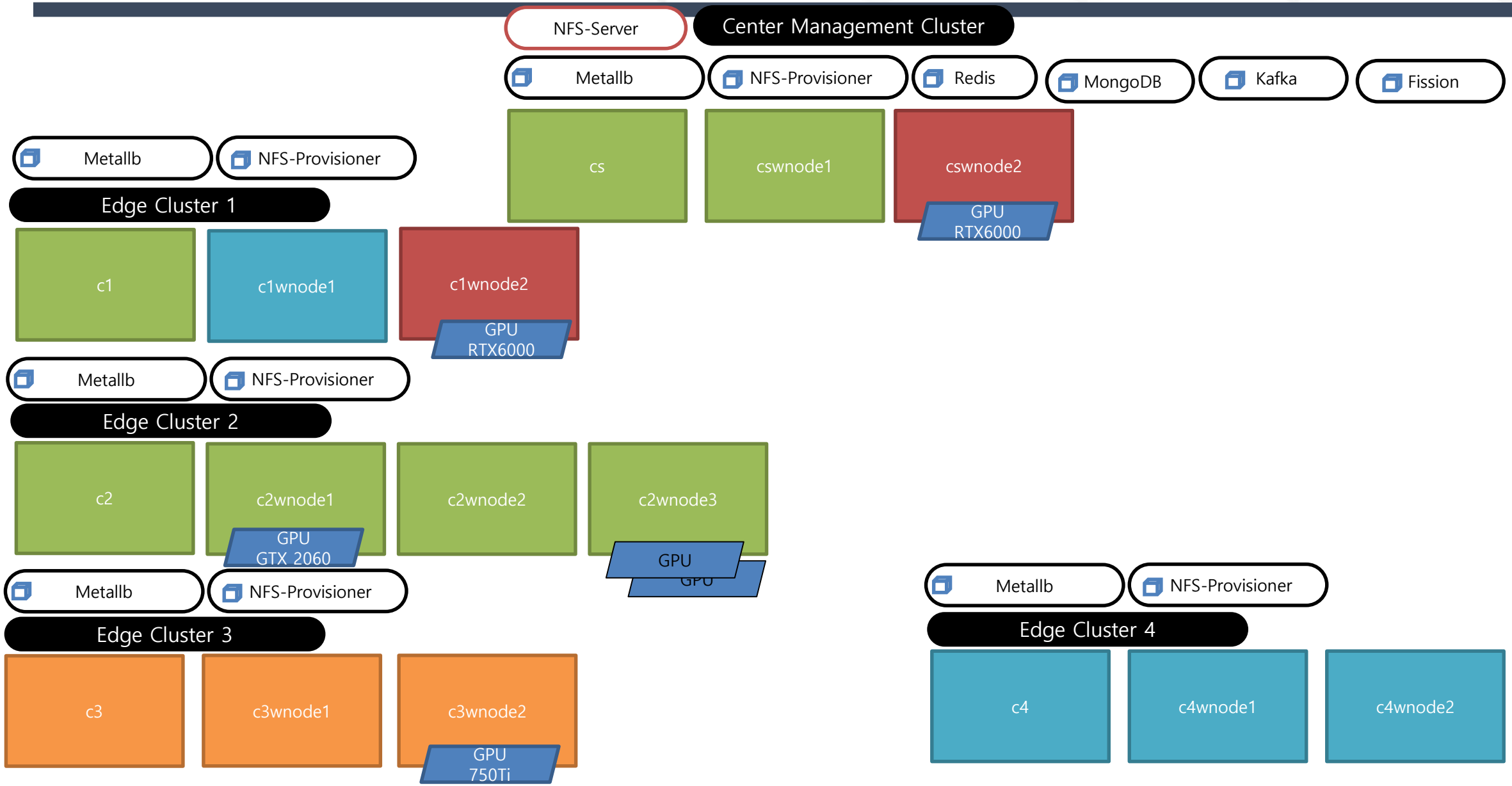
2 GS-Scheduler 핵심 기술

SelectClusters Policy



Ⅲ GS-Scheduler 공인인증 및 성능평가

3 GS-Scheduler 공인인증 및 성능평가

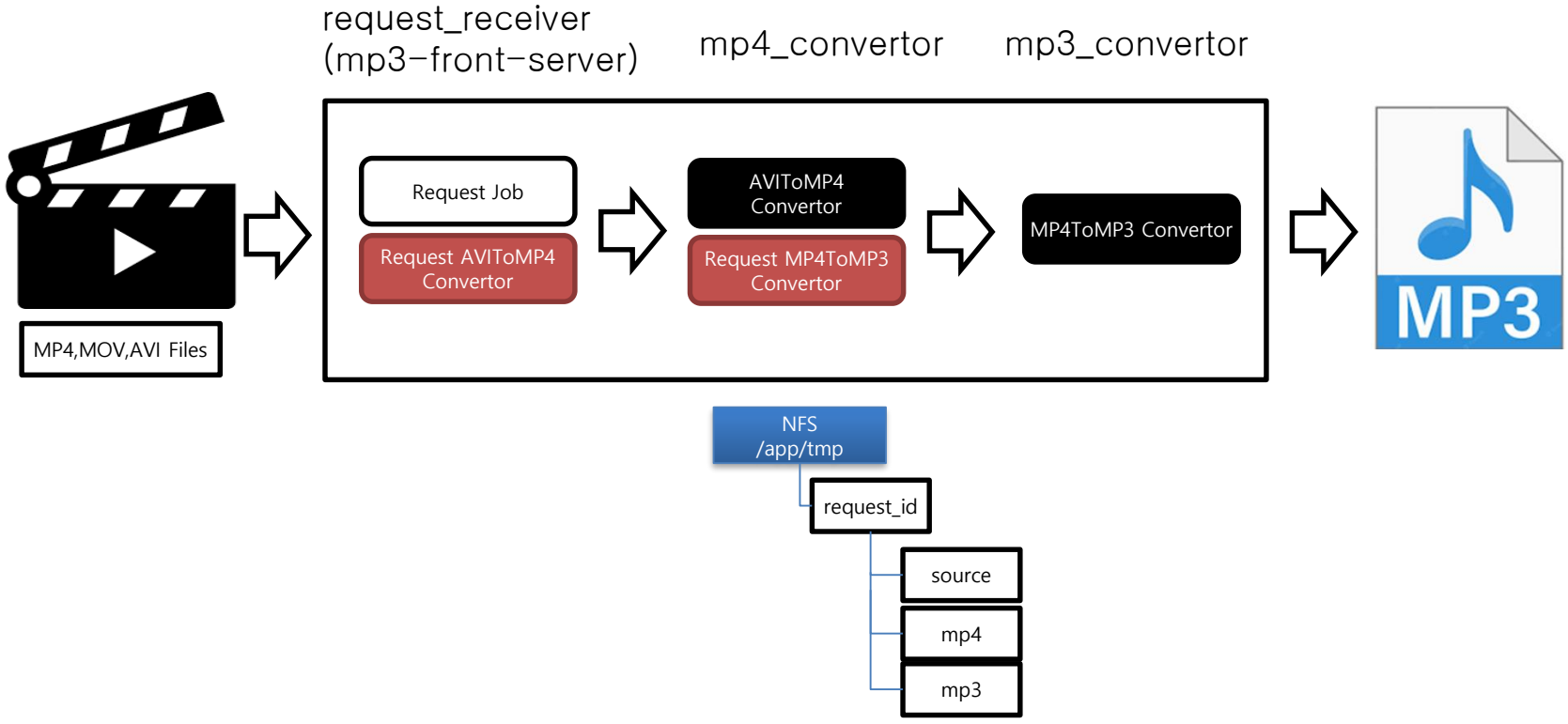


3 GS-Scheduler 공인인증 및 성능평가

클라우드 엣지 실행 환경 지원 수 3종 (Mono/Micro Services/FaaS)

AVI2MP3 Convertor

1 Mono Version



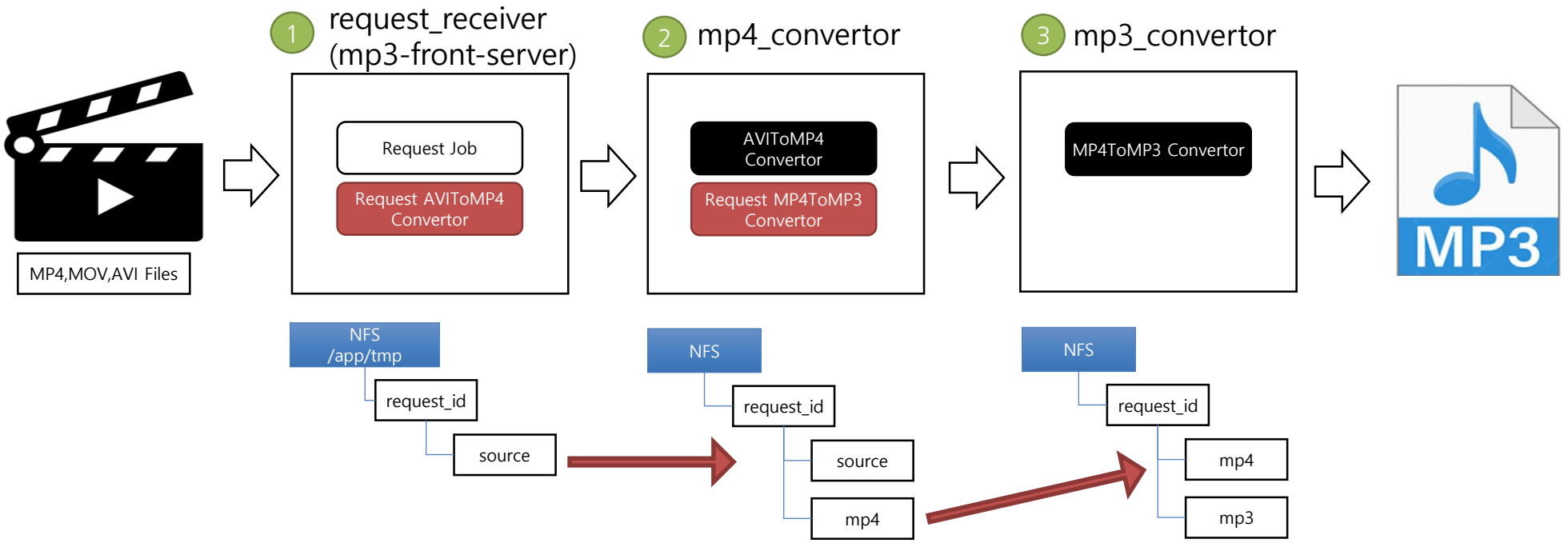
3 GS-Scheduler 공인인증 및 성능평가

GEdge Platform

클라우드 엣지 실행 환경 지원 수 3종

AVI2MP3 Convertor

2 Micro Version

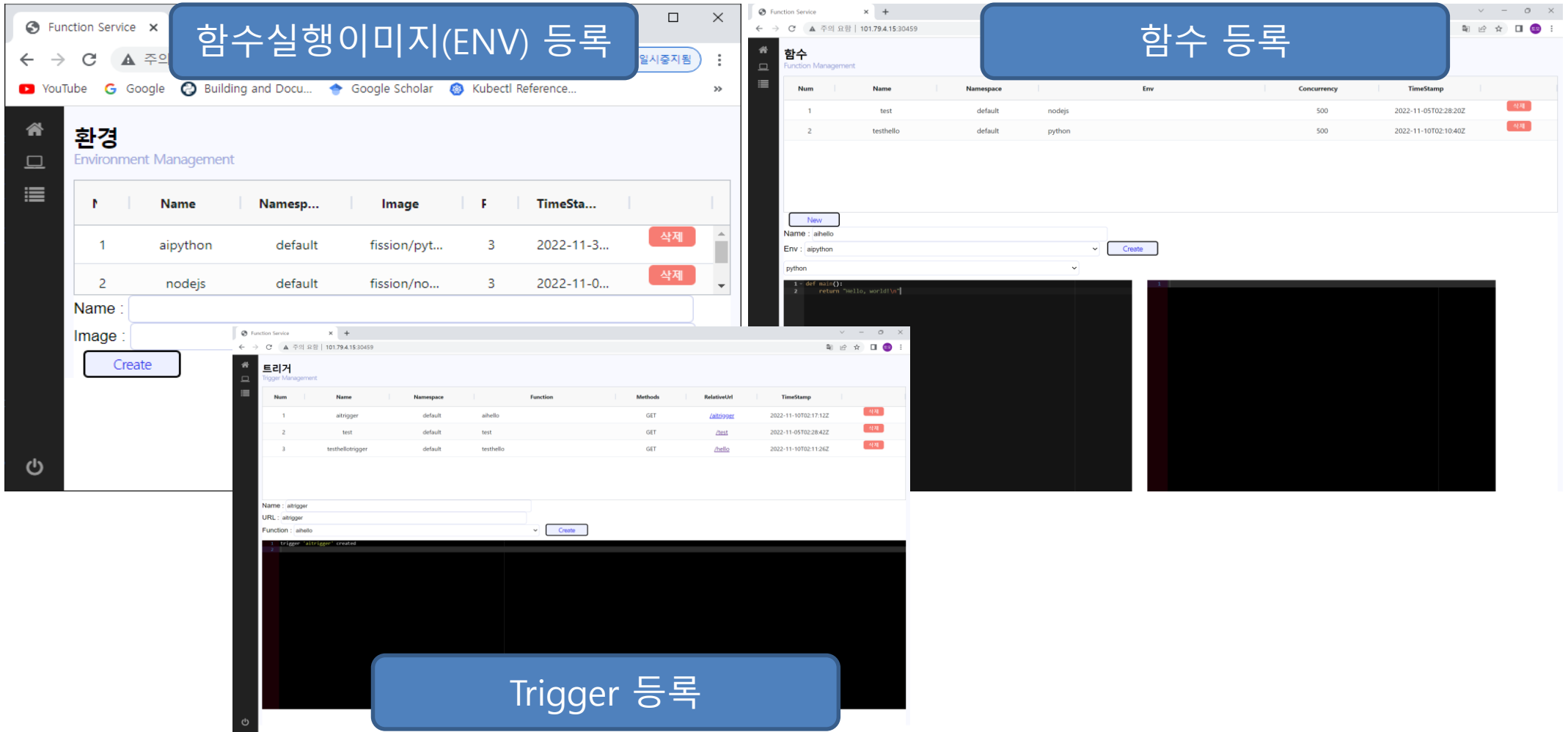


3 GS-Scheduler 공인인증 및 성능평가

GEdge Platform

클라우드 엣지 실행 환경 지원 수 3종

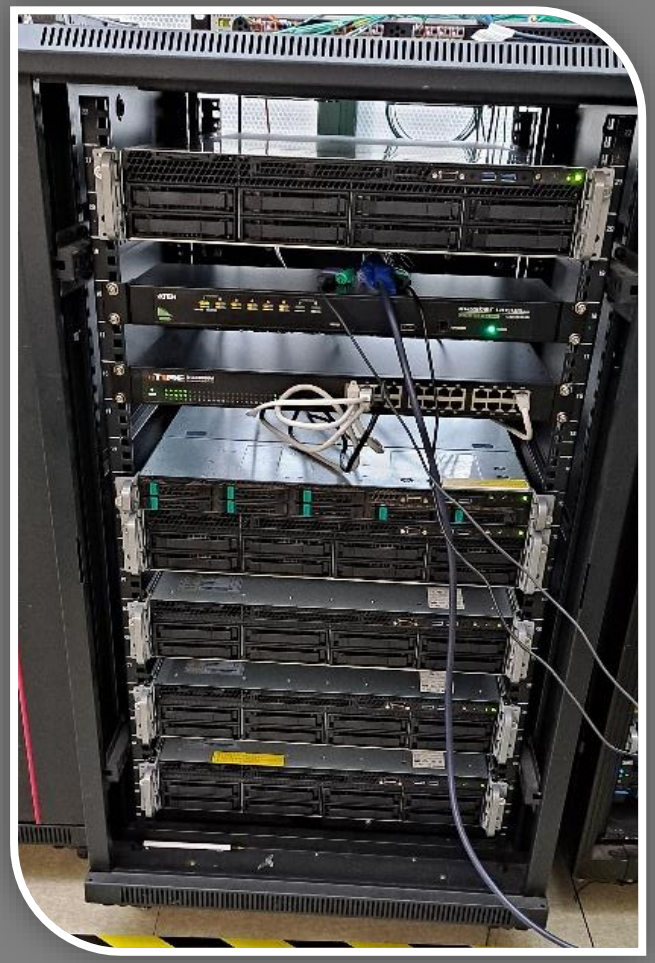
3 FaaS



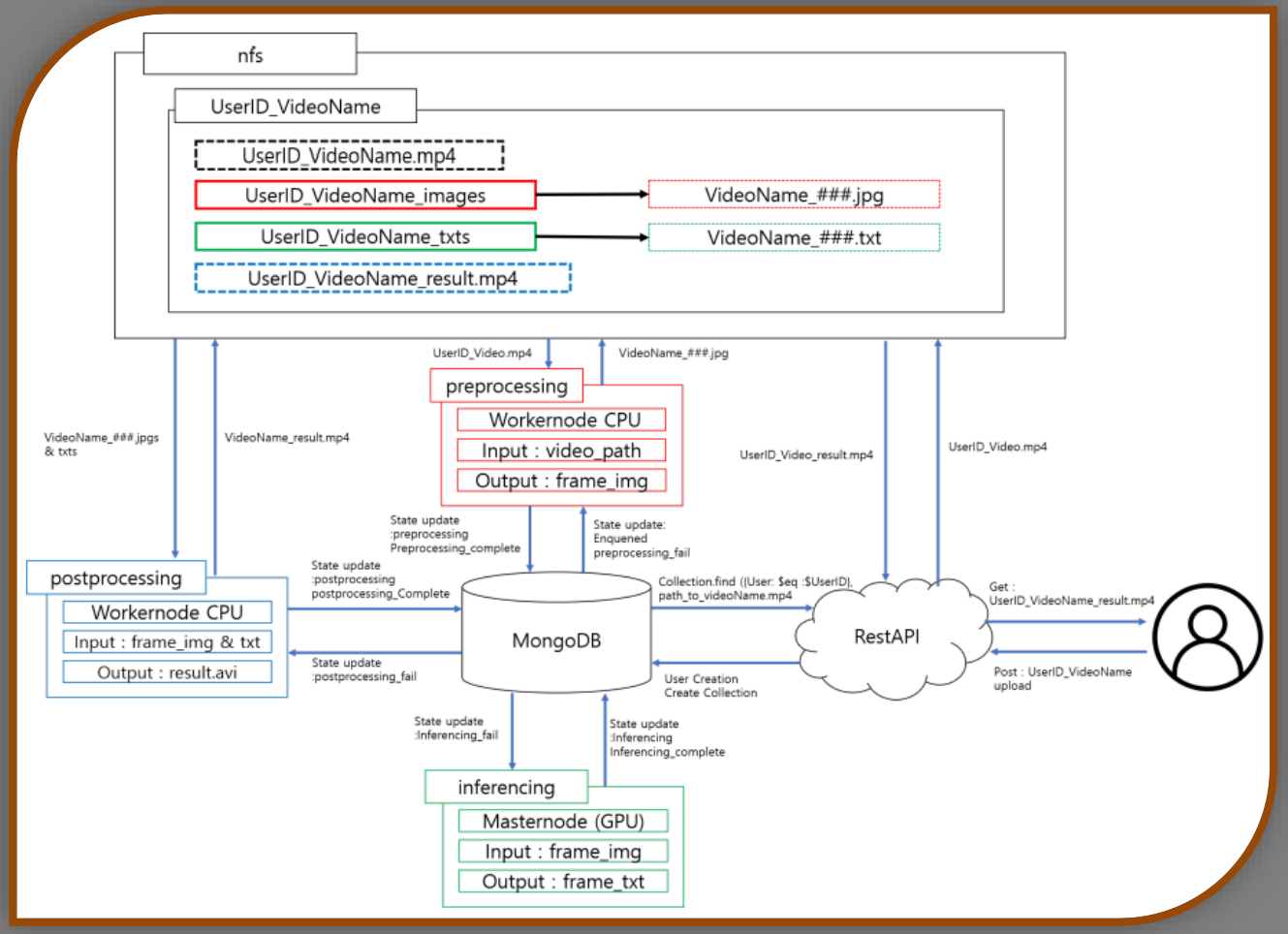
3 GS-Scheduler 공인인증 및 성능평가

GEdge Platform

클러스터 구성



객체 인식 서비스인 Yolo4 마이크로 서비스



3 GS-Scheduler 공인인증 및 성능평가

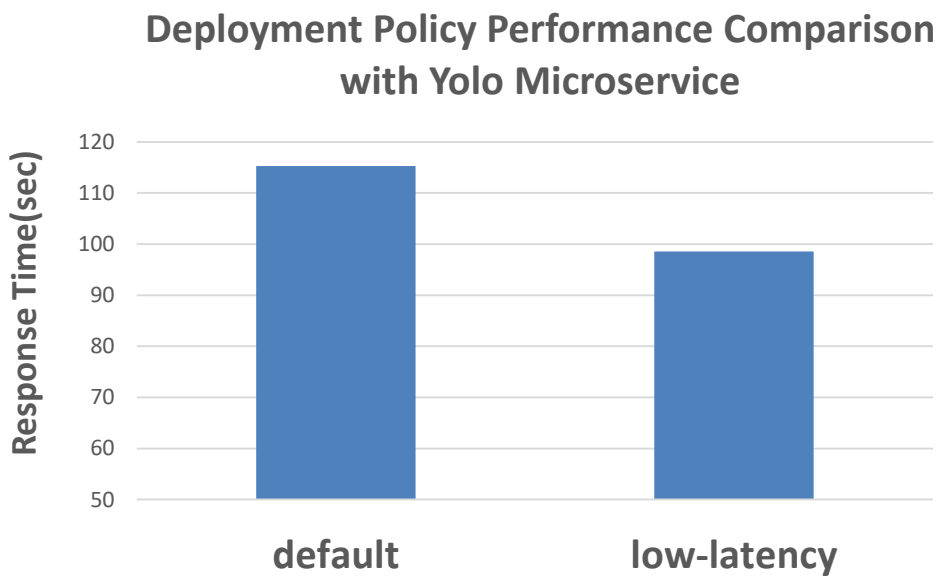
마이크로 서비스 배포정책의 성능평가

지능형 객체 인식 서비스 Yolo4로 검증

◆ 제안하는 마이크로 서비스 배포 정책의 성능 평가
- 워커 노드가 4개, 2개인 클러스터 2개를 구성
- **3개의 마이크로 서비스들로(선처리, 추론, 후처리) 구성된 지능형 객체 인식 서비스인 Yolo4 서비스를 자체 개발하여 평가**
- Yolo4 서비스의 주요 처리과정은 샘플 동영상 (30Mbyte)을 입력 값으로 받아서 추론을 통하여 객체인식 내용이 추가된 결과 동영상을 생성하는 방식

◆ 주요 테스트 서버 환경
- Xeon 3.0Ghz, Memory 1.1TB, GPU V100가 장착된 워커 노드
- 워커 **노드별 네트워크 지연시간의 차등**을 주기 위하여 네트워크 트래픽 부하 발생시켜서 특정 마스터 노드로부터 워커 노드 4개 (약 5ms), 2개(약 1ms)되도록 설정

◆ 성능평가 결과
- 제안하는 Low-Latency 부분이 적용하지 않은 것에 비하여 **약 20% 정도의 처리 속도가 향상**
- 노드들 간에 네트워크 지연시간 차이가 클수록 본 논문에서 제안하는 서비스 배포 정책이 더 의미가 있음



Ⅳ GS-Scheduler 개발 계획

4 GS-Scheduler 개발 계획

GEdge Platform

● User/Workspace/Project 개념을 추가한 GS-Scheduler 개발 및 고도화

users_Info

```
{
 user_name : u1,
 user_type : super/normal
 user_passwd : p1,
 user_email : e1,
 user_status: s1,
 workspace : ws1,
 projects: [p1,p2]
 }
{
 user_name : u2,
 user_type : super/normal
 user_passwd : p2,
 user_email : e2,
 user_status: s1,
 workspace : ws1,
 projects: [p3,p4]
 }
```

workspaces_Info

```
{
workspace_name: ws1,
selected_clusters : [ c1, c2 ],
cdate : cd1,
workspace_status: s1,
projects: [p1,p2]
}
{
workspace_name: ws2,
selected_clusters : [c2,c3],
cdate : cd1,
workspace_status: s1,
projects: [p1,p2]
}
```

projects_Info

```
{
  project_id : 'ws1: p1'
  workspace_name: ws1
  project_name : p1,
  user_name : u1
  selected_clusters: [c1,c2],
  cdate: cd1,
  project_status: s1
 }
{
  project_id : 'ws1: p2'
  workspace_name: ws1
  project_name : p2,
  user_name : u1
  selected_cluster: [c1,c2],
  cdate: cd2,
  project_status: s1
 }
```

4 GS-Scheduler 개발 계획

GEdge Platform

POD.Yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-test-resource
spec:
  containers:
  - name: nginx
    image: nginx
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
        nvidia.com/gpu: 1
```

pending_object

```
'spec': {'active_deadline_seconds': None,
         'affinity': None,
         'automount_service_account_token': None,
         'containers': [{'args': None,
                         'command': None,
                         'env': [{'name': 'gschConfig',
                                  'value': '{"type": "local", "priority": '
                                           '"GLowLatencyPriority", "option": '
                                           '{"sourceNode": "cswnode1"}}',
                                  'value_from': None}],
                         'env_from': None,
                         'image': 'nginx',
                         'image_pull_policy': 'Always',
                         'lifecycle': None,
                         'liveness_probe': None,
                         'name': 'nginx',
                         'ports': None,
                         'readiness_probe': None,
                         'resources': {'limits': {'cpu': '500m',
                                                  'memory': '128Mi',
                                                  'nvidia.com/gpu': '1'},
                                       'requests': {'cpu': '250m',
                                                    'memory': '64Mi',
                                                    'nvidia.com/gpu': '1'}},
```

다양한 필터 적용 개발 및 고도화
- Labels, Node, Volume, GPU, Memory, etc

get_request_filter_from_pending_object()

result_nodes=available_nodes(**request_filter**)

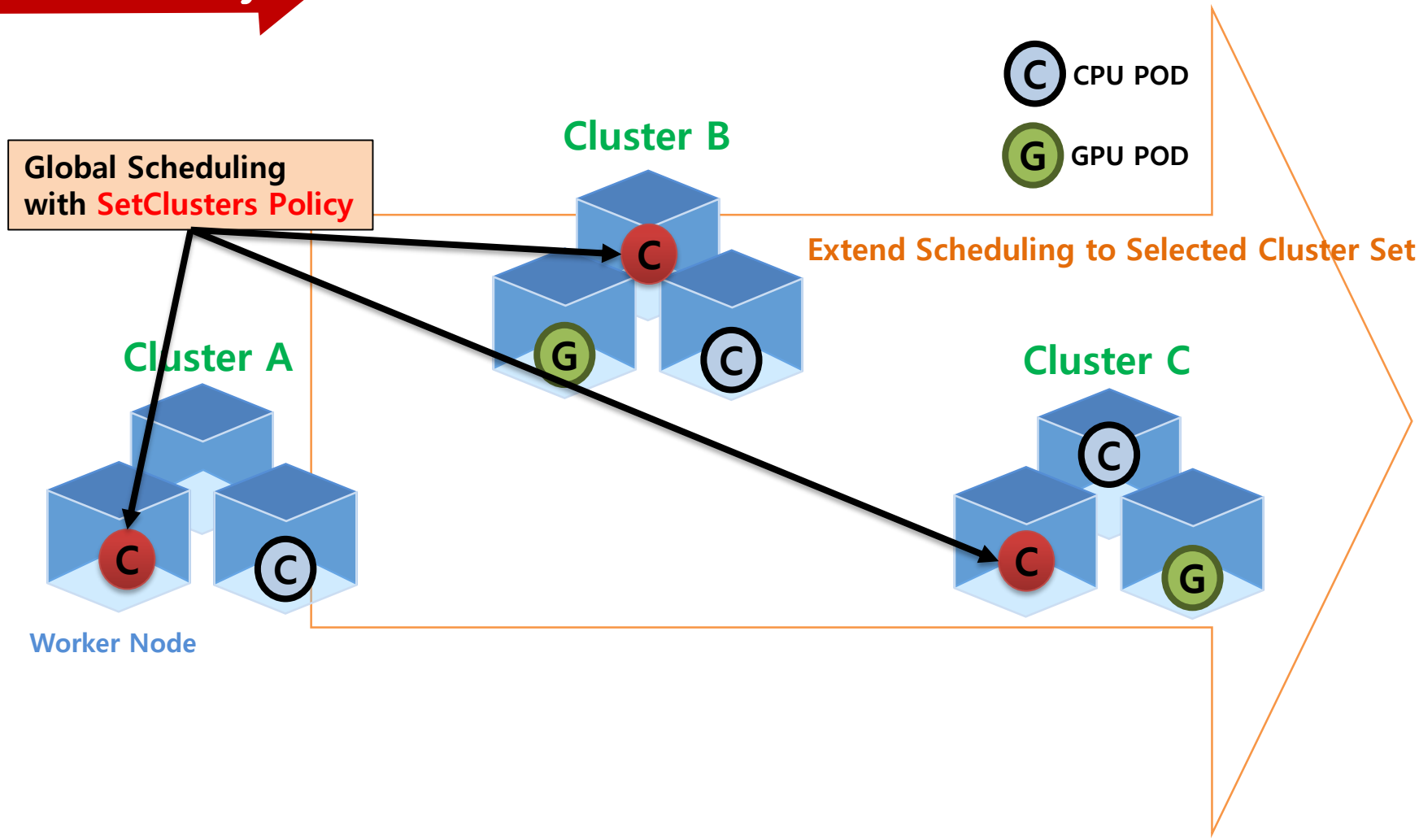
node_binding(**result_nodes**)

nodes_info

```
'status': {'addresses': [{'address': '129.254.202.47', 'type': 'InternalIP'},
                         {'address': 'cswnode2', 'type': 'Hostname'}],
           'allocatable': {'cpu': '96',
                           'ephemeral-storage': '194052659698',
                           'hugepages-1Gi': '0',
                           'hugepages-2Mi': '0',
                           'memory': '394792296Ki',
                           'nvidia.com/gpu': '1',
                           'pods': '110'},
           'capacity': {'cpu': '96',
                        'ephemeral-storage': '210560612Ki',
                        'hugepages-1Gi': '0',
                        'hugepages-2Mi': '0',
                        'memory': '394894696Ki',
                        'nvidia.com/gpu': '1',
                        'pods': '110'},
```

2 GS-Scheduler 핵심 기술

SetClusters Policy

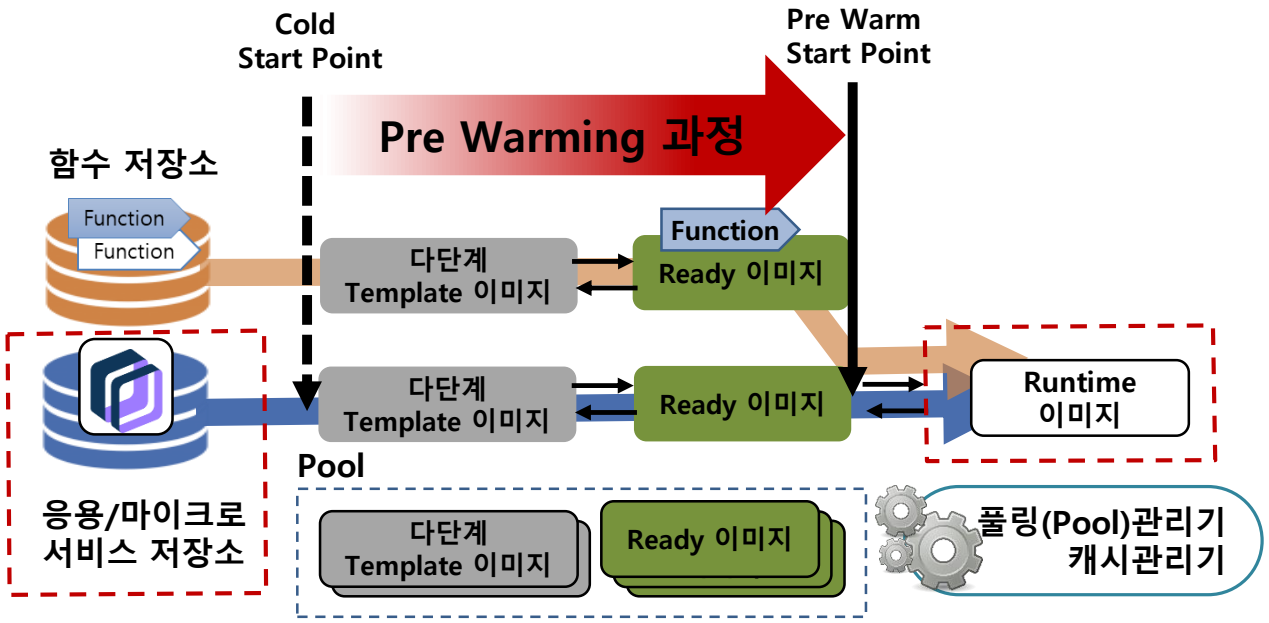


4 GS-Scheduler 개발 계획

- GS-Scheduler 설치 고도화
- Policy별 스케일링 제어기 개발 및 고도화
- Job / DemonSet / RestfulSet 다양한 리소스 할당 적용
- 글로벌 스케줄링에서 Storage Class / Volume 리소스 고려 적용
- 3가지 실행 형태(마이크로 서비스/비 마이크로 서비스/함수)의 특성을 고려한 자원 할당 스케줄러 기술 최적화

4 GS-Scheduler 개발 계획

- 이벤트 발생에 따른 함수 자동 실행기 핵심 기술 설계 및 개발
- 3종 환경에 실행 가능한 함수 실행 기술 최적화 및 통합 검증
- 엣지 응용의 고속 실행을 위한 컨테이너 풀링(Pooling)기반 2레벨 실행 (Pre-warm start & Warm start) 기술 최적화 및 통합



감사합니다.

http://gedge-platform.github.io

GS-Scheduler 코어개발자
발표자 장수민(jsm@etri.re.kr)