



GEdge Platform

GEdge(Griffin-Edge) Platform

- 초저지연 지능형 클라우드 엣지 SW 플랫폼 -

강화학습 기반 멀티엣지 협업 정책 생성 기술

2021.12.09

GS-Link 프레임워크 코어 개발자 (GS-Linkhq)

윤주상 (joosang.youn@gmail.com)

“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

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)

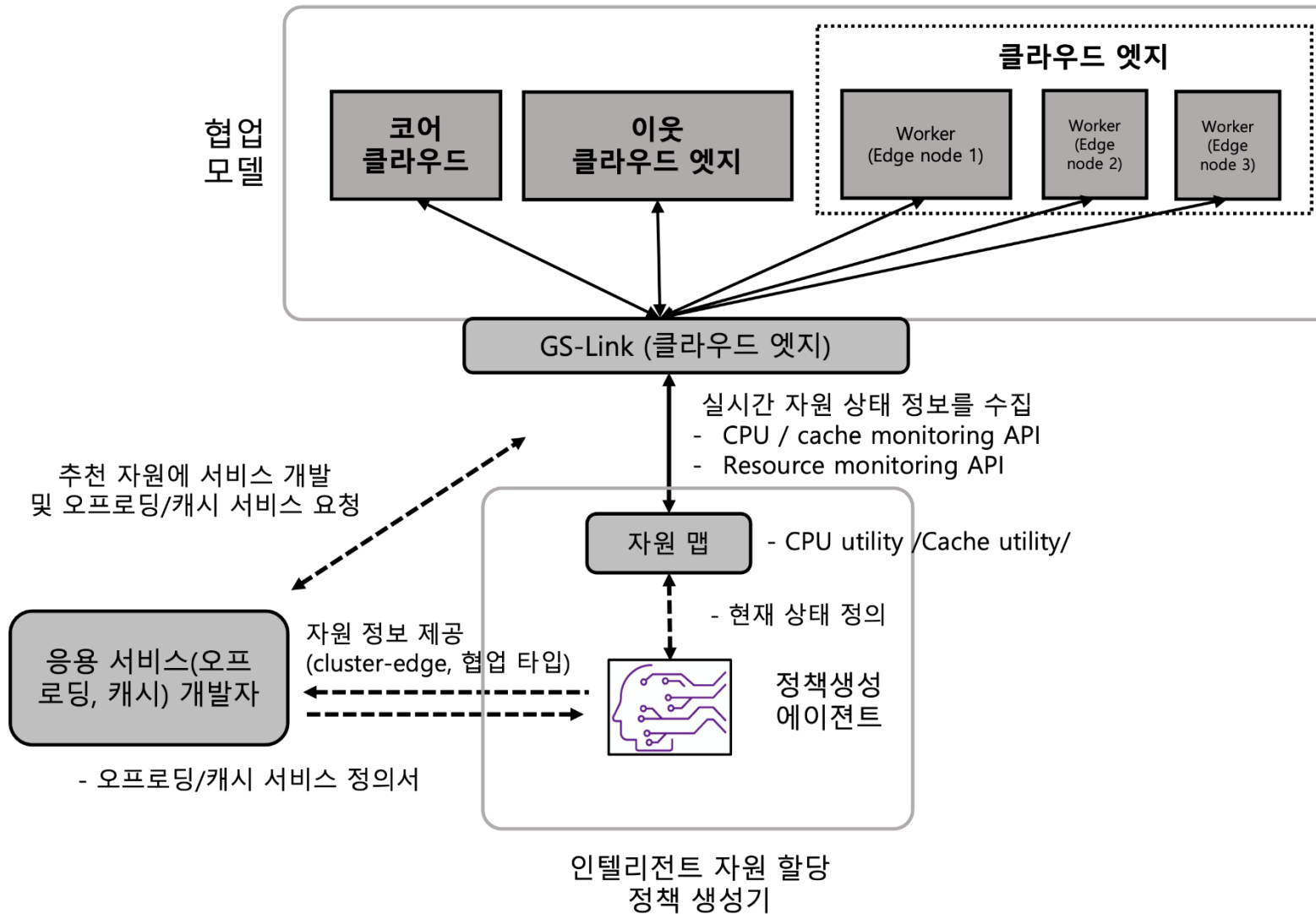
클라우드 엣지 서비스 플랫폼 (GS : GEdge Service Platform)

Contents

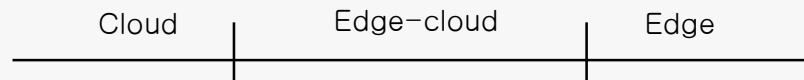
- I 강화학습 기반 지능형 오프로딩 정책 생성 기술
- II 지능형 서비스 이동 정책 기술
- III 지능형 협업 캐시 정책 기술
- IV 향후 연구

강화학습 기반 지능형 오프로딩 정책 생성 기술

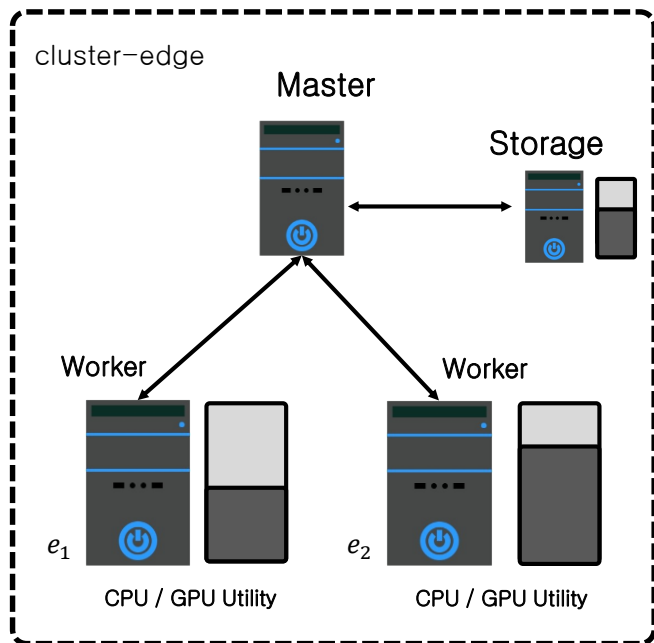
1 정책 생성 및 제공 과정



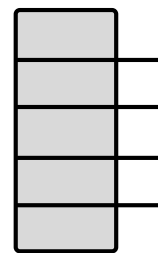
- 자원 할당을 위한 정책
 - Random
 - Least Load : 자원 사용률이 가장 낮은 곳
 - Round-Robin : 순서에 따라서 자원 제공
 - Rule : 수직 협업 경우 사용률에 따라 클라우드 자원 추천
 - DRL : 심층강화학습 기반 정책 생성
- 수직적 협업 종류 (Rule 정책 적용)
 - Edge
 - 요구 자원 (20) < 허용 가능 자원 ($20 + (20 \times 0.1) = 22$) 이상
 - Edge-Cloud
 - 허용 가능 자원 ($20 - (20 \times 0.3) = 16$) < 요구 자원 (20) < 허용 가능 자원 ($20 + (20 \times 0.1) = 22$) 이상
 - Cloud
 - 요구 자원 (20) > 허용 가능 자원 ($20 - (20 \times 0.3) = 16$)



- Cluster-edge computation resource modeling



워커 노드 자원
상태 정의 (5-level)

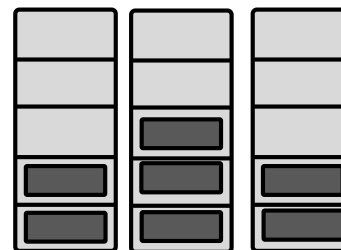


워커 노드 자원 상태
 $S_1 = \{0, 1, 2, 3, 4\}$

$S(t) = 1 \rightarrow$

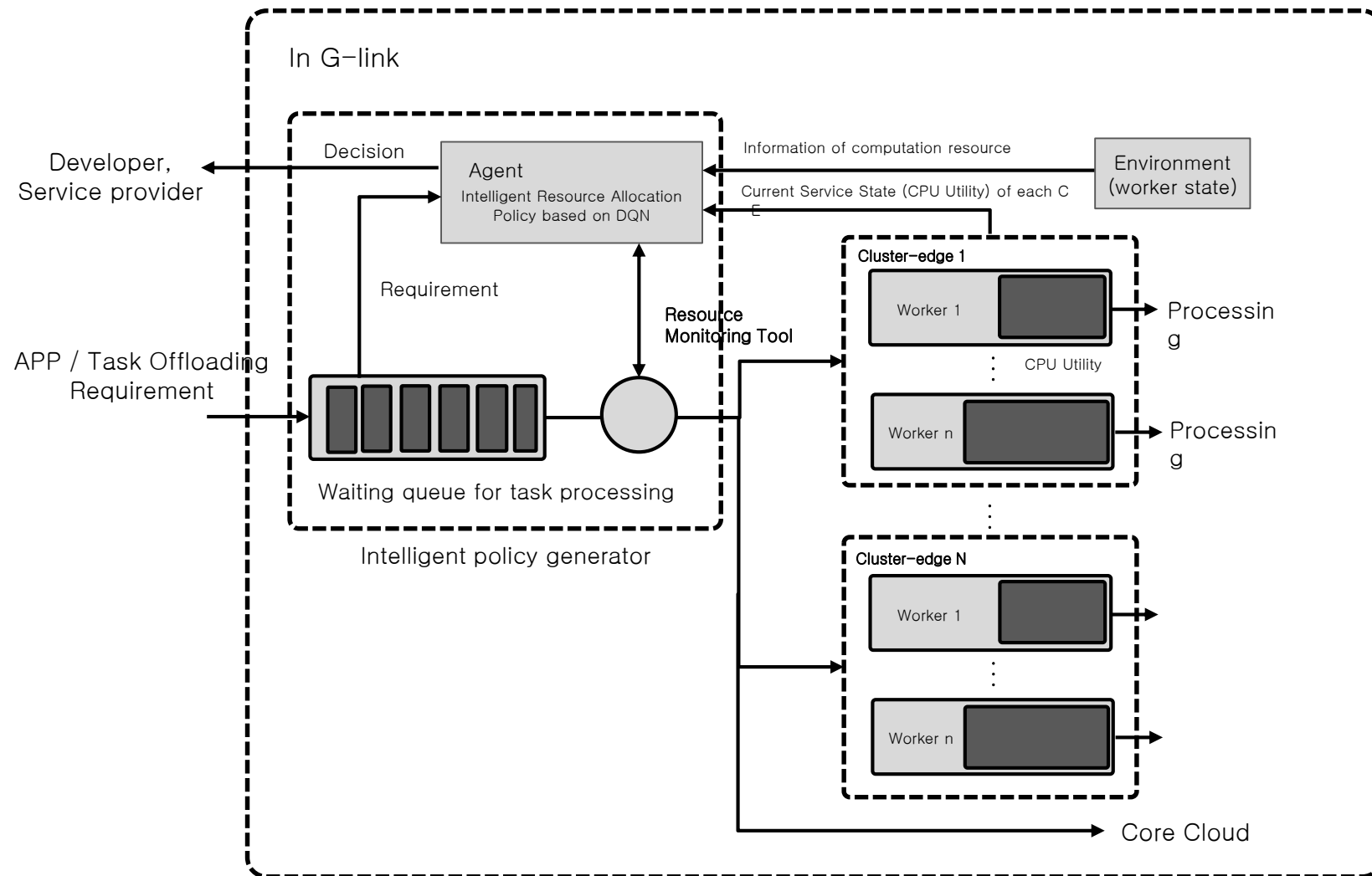


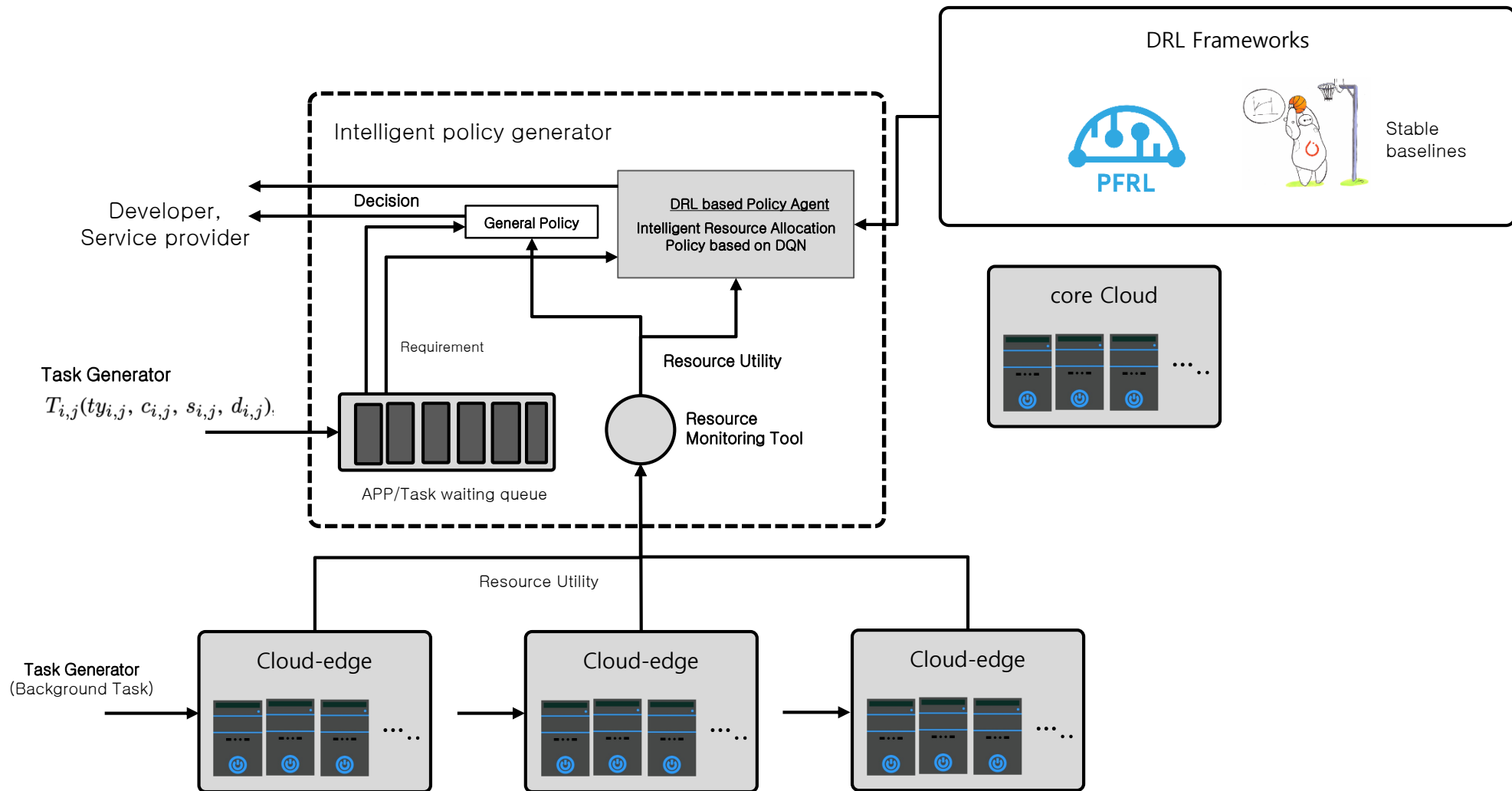
자원 상태 모델
3 workers 경우

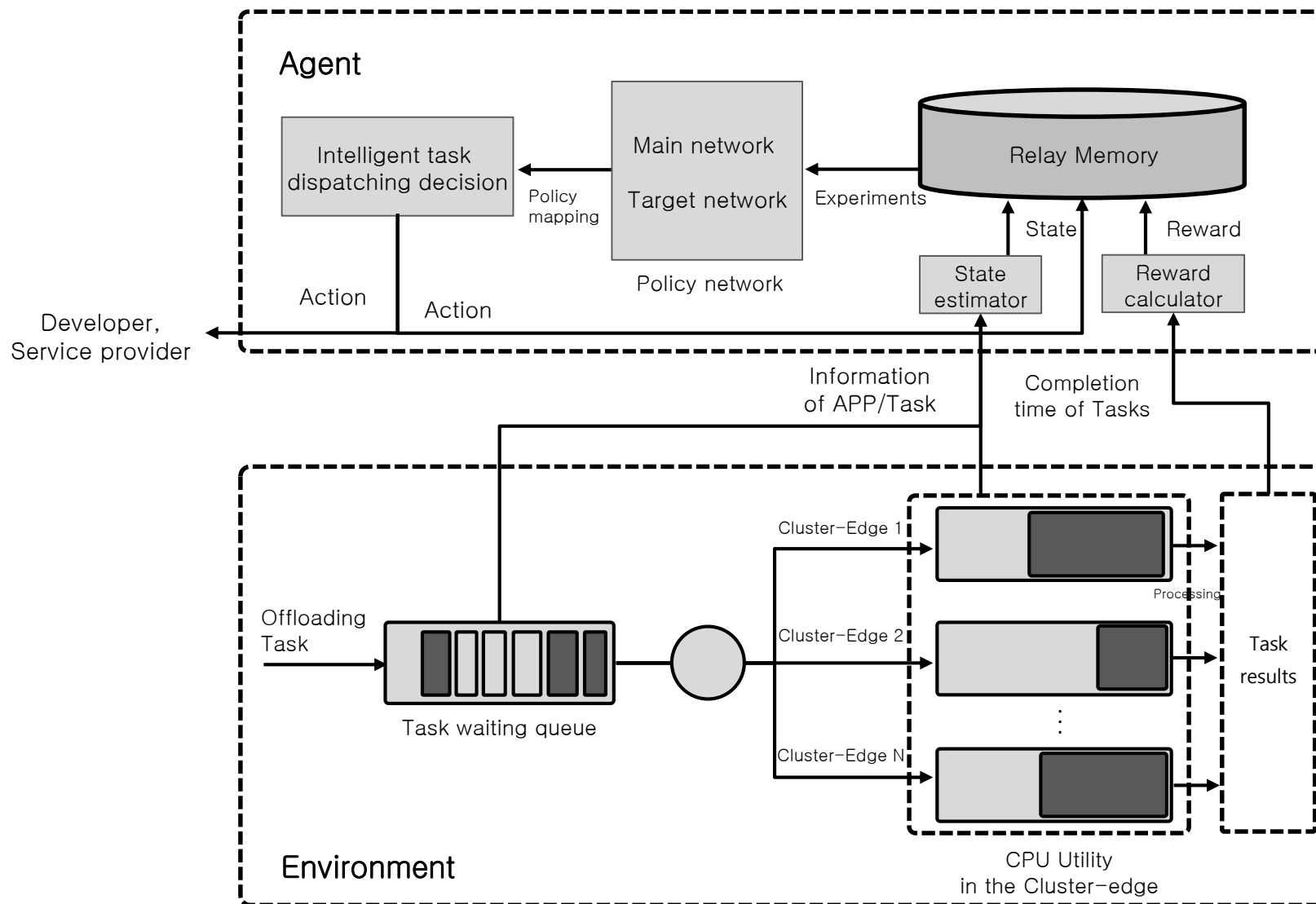


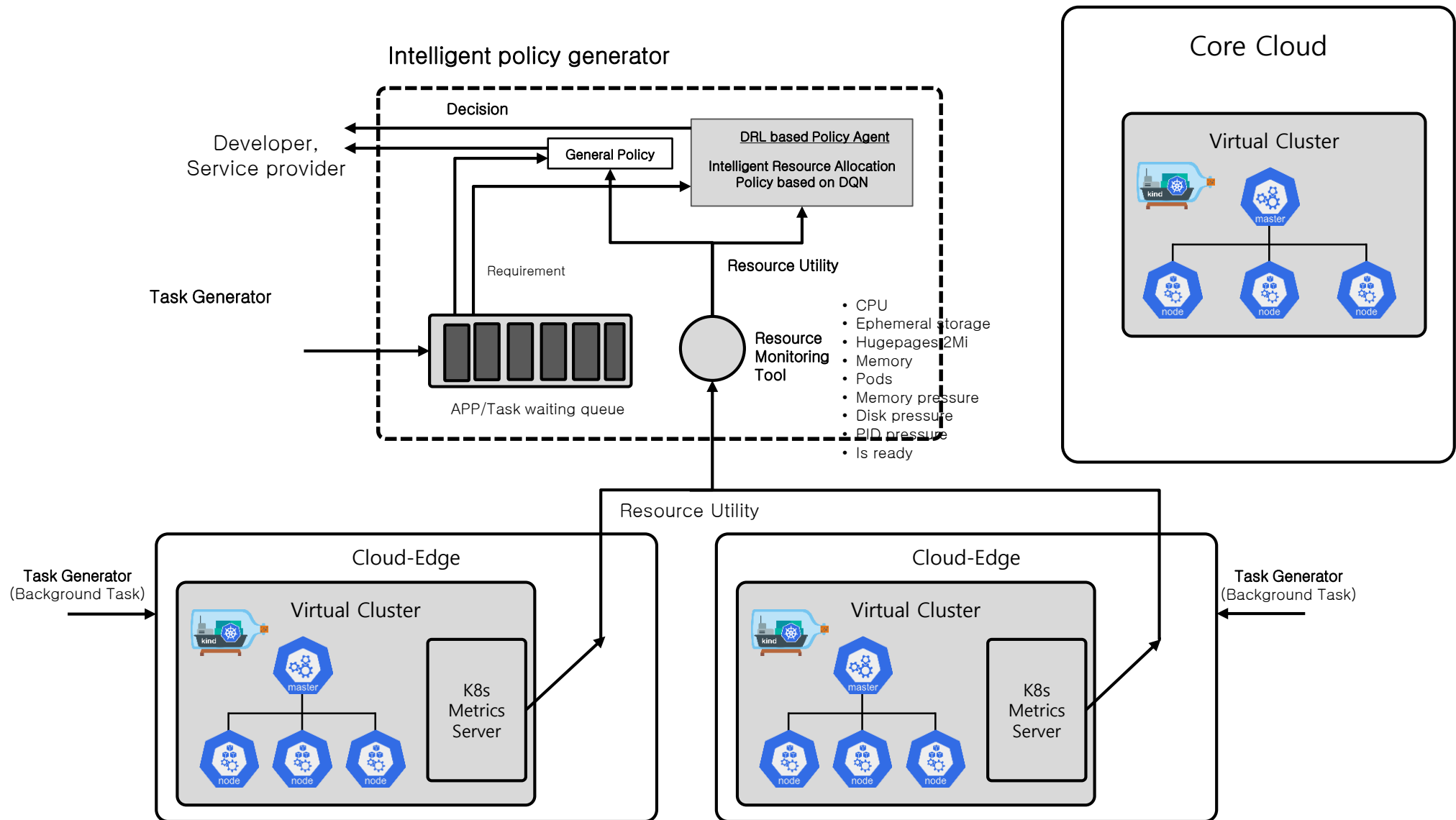
$S = \{0, 1, 2, 3, 4, \dots N\}$
 $N = 5 \times 5 \times 5 = 125 - 1 = 124$

$S(t) = (S_1, S_2, S_3)$
 $= (0, 0, 0) = 0$
 $= (0, 0, 1) = 1$
 $= (0, 0, 2) = 2$
 \dots









Source Code

```
...
class Cluster:
    def __init__(self, name, ip):
        self.name = name
        self.ip = ip
        self.nodes = {}

    def request_status(self):
        port = '8080'
        try:
            url = 'http://' + self.ip + port
            url += '/api/v1/nodes'
            r = requests.get(url)

            if r.ok:
                return r.json()

        except requests.exceptions.ConnectionError as e:
            print("Error:", e)
            return False

    def update_status(self):
        status = self.request_status()
        if status:
            for item in status['items']:
                node_name = item['metadata']['name']

                if node_name not in self.nodes:
                    self.nodes[node_name] = Node(node_name)

                capacity = item['status']['capacity']
                allocatable = item['status']['allocatable']
                conditions = item['status']['conditions']

                cpu_capacity = capacity['cpu']
                cpu_allocatable = allocatable['cpu']
                ephemeral_storage_capacity = capacity['ephemeral-storage']
                ephemeral_storage_allocatable = allocatable['ephemeral-storage']
                hugepages_2Mi_capacity = capacity['hugepages-2Mi']
                hugepages_2Mi_allocatable = allocatable['hugepages-2Mi']
                memory_capacity = capacity['memory']
                memory_allocatable = allocatable['memory']
                pods_capacity = capacity['pods']
                pods_allocatable = allocatable['pods']

                has_memory_pressure = None
                has_disk_pressure = None
                has_pid_pressure = None
                is_ready = None

                for condition in conditions:
                    if condition['type'] == 'MemoryPressure':
                        has_memory_pressure = (condition['status'] == 'True')
                    if condition['type'] == 'DiskPressure':
                        has_disk_pressure = (condition['status'] == 'True')
                    if condition['type'] == 'PIDPressure':
                        has_pid_pressure = (condition['status'] == 'True')
                    if condition['type'] == 'Ready':
                        is_ready = (condition['status'] == 'True')

...

```

Example Output

```
Cluster name: cluster1

Node name: kind-control-plane
CPU(capacity/allocatable): 6/6
Ephemeral storage(capacity/allocatable): 28768292Ki/28768292Ki
Hugepages 2Mi(capacity/allocatable): 0/0
Memory(capacity/allocatable): 8047092Ki/8047092Ki
Pods(capacity/allocatable): 110/110
Memory pressure: False
Disk pressure: False
PID pressure: False
Is ready: True

Node name: kind-worker
CPU(capacity/allocatable): 6/6
Ephemeral storage(capacity/allocatable): 28768292Ki/28768292Ki
Hugepages 2Mi(capacity/allocatable): 0/0
Memory(capacity/allocatable): 8047092Ki/8047092Ki
Pods(capacity/allocatable): 110/110
Memory pressure: False
Disk pressure: False
PID pressure: False
Is ready: True

Node name: kind-worker2
CPU(capacity/allocatable): 6/6
Ephemeral storage(capacity/allocatable): 28768292Ki/28768292Ki
Hugepages 2Mi(capacity/allocatable): 0/0
Memory(capacity/allocatable): 8047092Ki/8047092Ki
Pods(capacity/allocatable): 110/110
Memory pressure: False
Disk pressure: False
PID pressure: False
Is ready: True

Node name: kind-worker3
CPU(capacity/allocatable): 6/6
Ephemeral storage(capacity/allocatable): 28768292Ki/28768292Ki
Hugepages 2Mi(capacity/allocatable): 0/0
Memory(capacity/allocatable): 8047092Ki/8047092Ki
Pods(capacity/allocatable): 110/110
Memory pressure: False
Disk pressure: False
PID pressure: False
Is ready: True

```

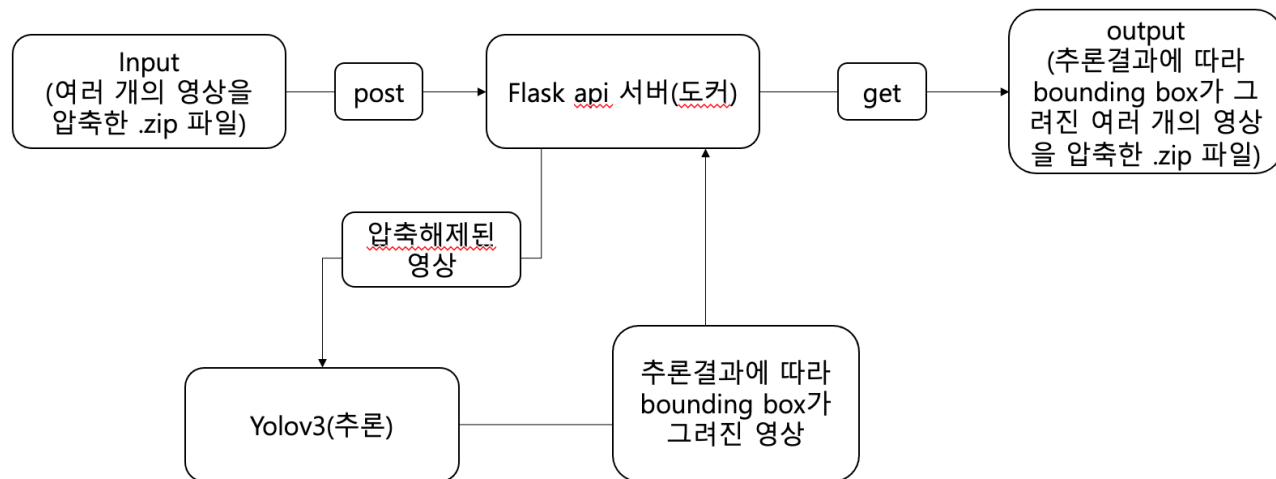
Example of Service Requirements

```
name: foo-service
config: foo-config.yaml
requirements:
  cpu: 260
  mem: 600
  storage: 1100
  tst: 100
  ...
```

Example of Service Configuration

```
kind: Pod
apiVersion: v1
metadata:
  name: foo-app
  labels:
    app: foo
spec:
  containers:
  - name: foo-app
    image: foo:1.0
    args:
    - -text=foo
---
kind: Service
apiVersion: v1
metadata:
  name: foo-service
spec:
  selector:
    app: foo
  ports:
  - port: 80
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example-ingress
spec:
  rules:
  - http:
      paths:
      - pathType: Prefix
        path: /foo
        backend:
          service:
            name: foo-service
            port:
              number: 80
      - pathType: Prefix
        path: /bar
        backend:
          service:
            name: bar-service
            port:
              number: 80
  ---
```

Task를 위한 서비스





```

@app.route('/predict', methods=['POST'])
def predict():
    # 전달받은 request에서 이미지 데이터 받고 byte로 변환
    file = request.files['file']
    bytes = file.read()
    with Zipfile(bytesIO(bytes)) as zipfile:
        zipfile.extractall('data')
    opt = parse_opt() # 추론 옵션 설정
    main(opt) # 추론 실행
    return 'OK', 200

@app.route('/result')
def result():
    zip_file = zipfile.ZipFile(os.path.join('runs', 'output.zip'), "w")
    for (path, dir, files) in os.walk('runs/detect'):
        for file in files:
            if not (file.endswith('.zip')):
                zip_file.write(os.path.join('runs/detect', file), compress_type=zipfile.ZIP_DEFLATED)
    zip_file.close()
    return send_file(os.path.join('runs', 'output.zip'))
  
```

추론전

추론후

 Vmmem	2.8%	32,045.7MB	0MB/s	0Mbps	0%
 Vmmem	62.3%	32,571.8MB	0MB/s	0Mbps	0%

- Objective function

$$\tilde{D}_{i,j} = \min \frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M D_{i,j}$$

- State function

$$S = \{s \mid s = (\hat{D}^{q^c}, D^{q_1^e}, D^{q_2^e}, \dots, D^{q_n^e})\},$$

- Action function

$$A = \{a \mid a = (e_1, e_2, \dots, e_n)\}$$

- Reward function

$$R(s(\tau), a(\tau)) = \begin{cases} \exp(\tilde{D}_{i,j} - D_{i,j}) & \text{if } \tilde{D}_{i,j} > D_{i,j} \\ -\exp(D_{i,j} - \tilde{D}_{i,j}) & \text{if } D_{i,j} > \tilde{D}_{i,j} \end{cases}$$

Algorithm 1 Task dispatching algorithm in the cluster-edge

Input: computing capability of edge nodes, radio bandwidth resource, parameters for the task setting

Output: edge node selection $a(\tau)$

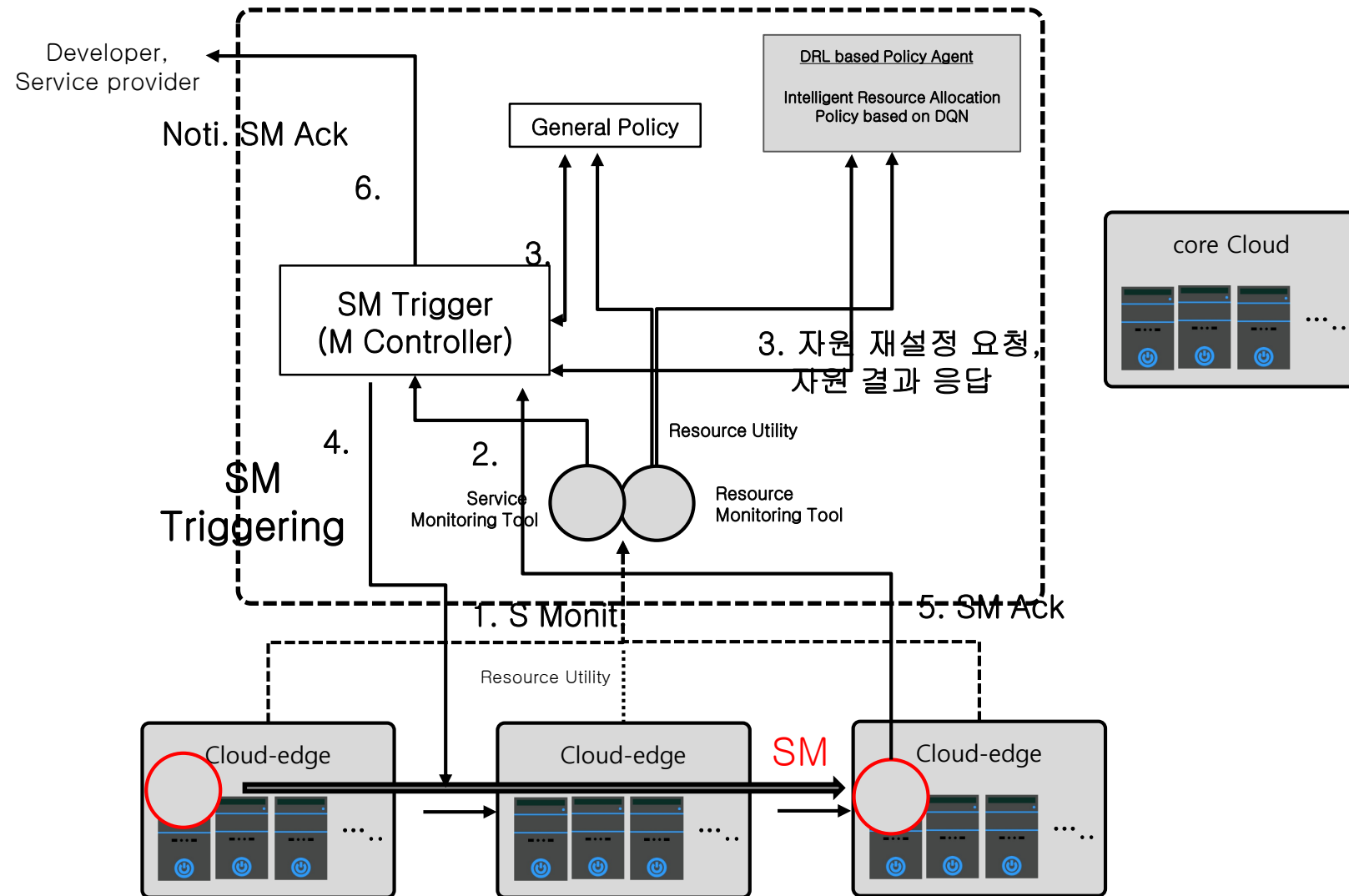
```

1: Initiate learning rate  $\sigma$ ;
2: Initiate  $\tau$ ;
3: Initiate the number of mini-batches  $B$ ;
4: Initiate batch size  $N$ ;
5: Initiate experience replay memory with max size  $K$ ;
6: Initiate main network  $Q$  with random parameter  $\theta$ ;
7: Initiate target network  $\tilde{Q}$  with parameter  $\tilde{\theta} = \theta$ ;
8: for episode  $e = 1 \dots MaxSteps$  do
9:   for  $\tau = 1 : T$  do
10:    Get the current state  $s_\tau$  from the environment;
11:    Take action
12:     $a(\tau) = \begin{cases} \text{random action from } A(s), \text{ prob. } \epsilon \\ \text{argmax}_{a \in A(s)} Q(s, a; \theta), 1 - \epsilon; \end{cases}$ 
13:    Perform  $a(\tau)$ , receive  $r(\tau)$  and perform
14:    state transition  $s(\tau) \rightarrow s(\tau + 1)$ ;
15:    Store experiences  $e = (s(\tau), a(\tau), r(\tau), s(\tau + 1))$ 
16:    using the current policy into ERM;
17:    for  $b = 1 \dots B$  do
18:      Randomly sample a mini-batch  $b$  of experiences
19:      from ERM
20:      for  $i = 1 \dots N$  do
21:        # Calculate target Q-values for each example
22:         $y_i = r_i + \delta_{s'_i} \gamma \max_{\tilde{a}_i} \tilde{Q}^\pi(s'_i, \tilde{a}_i; \tilde{\theta})$  where
23:         $\delta_{s'_i} = 0$  if  $s'_i$  is terminal, 1 otherwise;
24:      end for
25:      Calculate the loss  $L(\theta)$  by (24);
26:      Update the network's parameters  $\theta$  by (18);
27:      Set  $\tilde{Q} = Q$ ;
28:    end for
29:  end for
30:  Decay  $\tau$ 
31: end for
  
```



지능형 서비스 이동 정책 기술



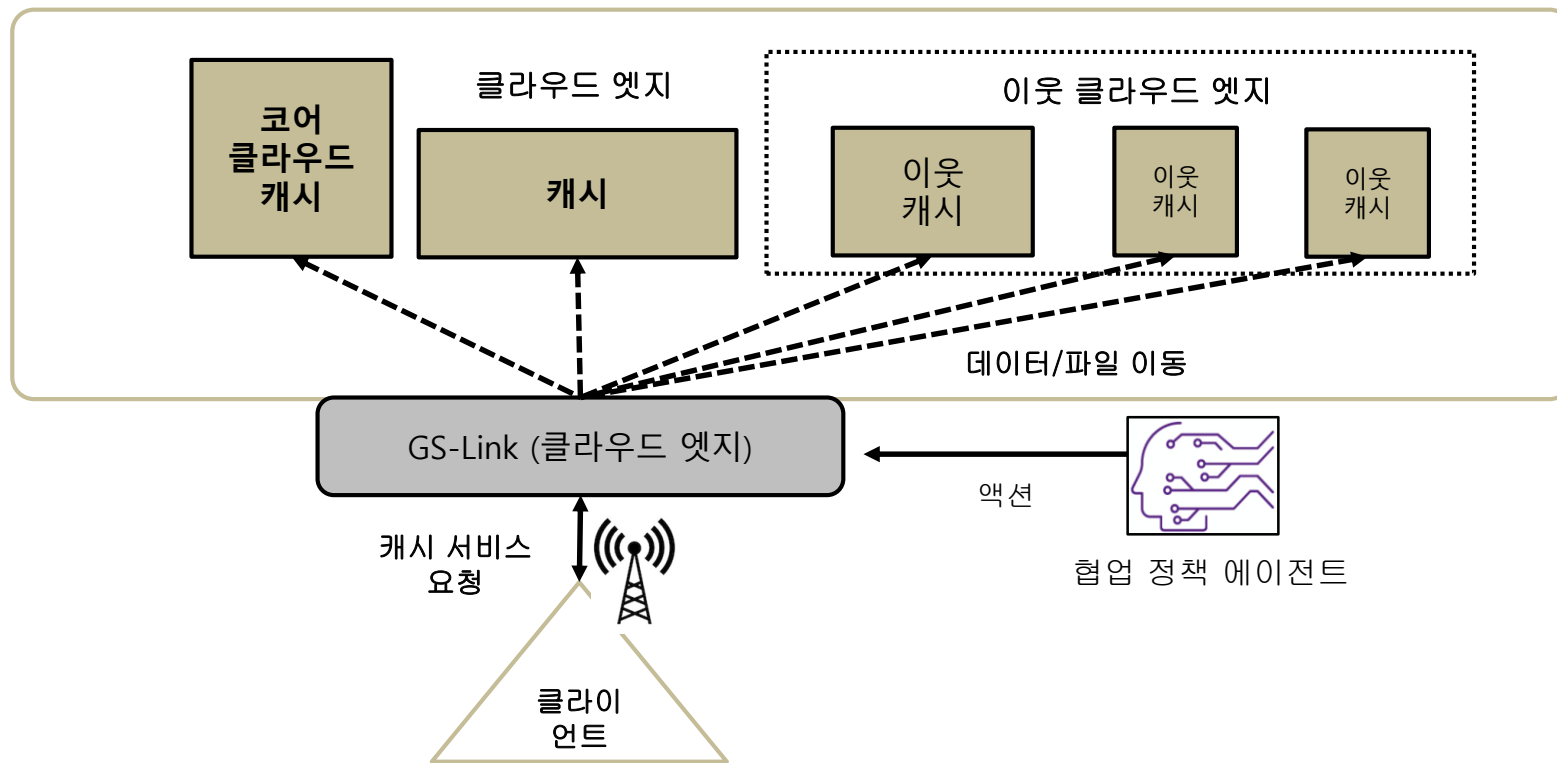




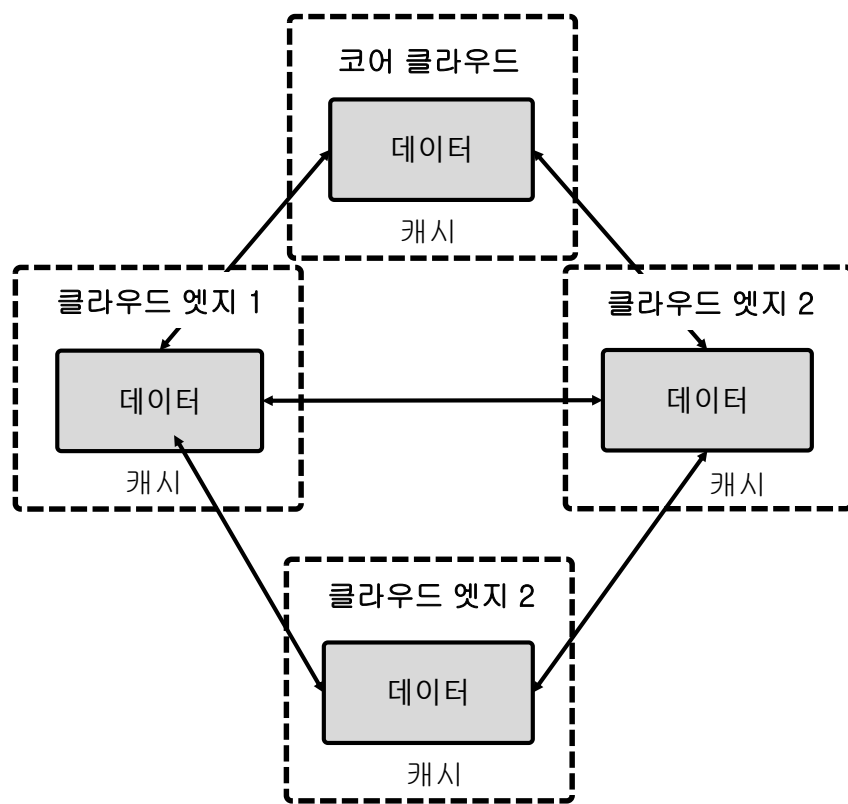
지능형 협업 캐시 정책 기술



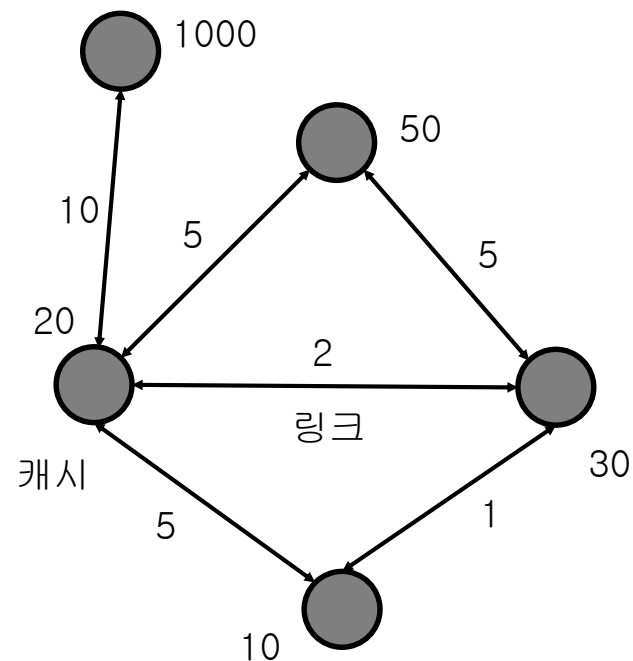
- 협업 캐시 모델 (추상적 모델)



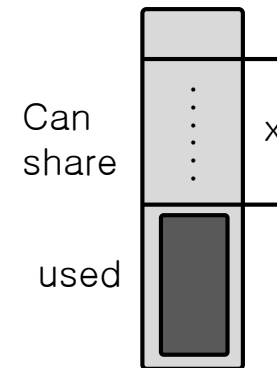
- 협업 캐시 물리적 네트워크



물리적 캐시 네트워크 모델

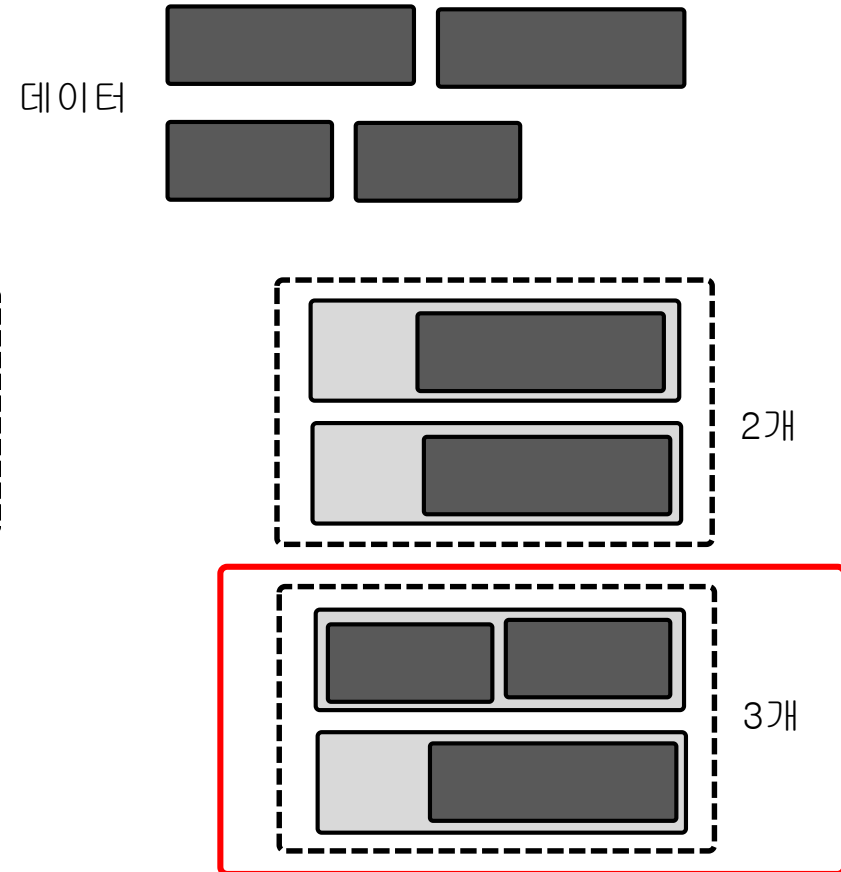
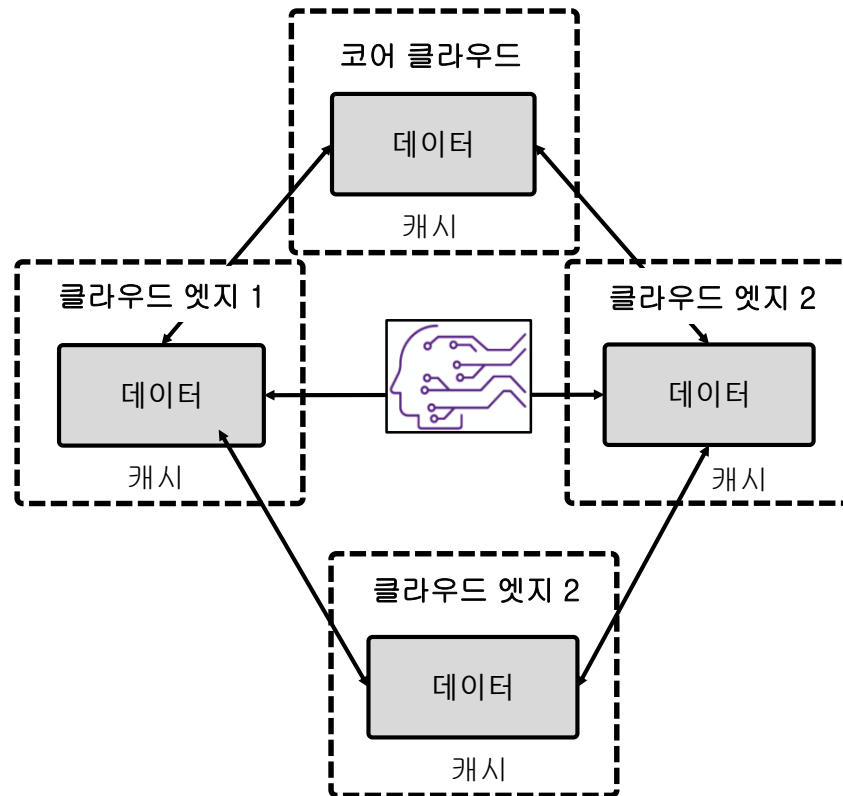


그래프 기반 캐시 네트워크

캐시 자원
정의 (x-level)

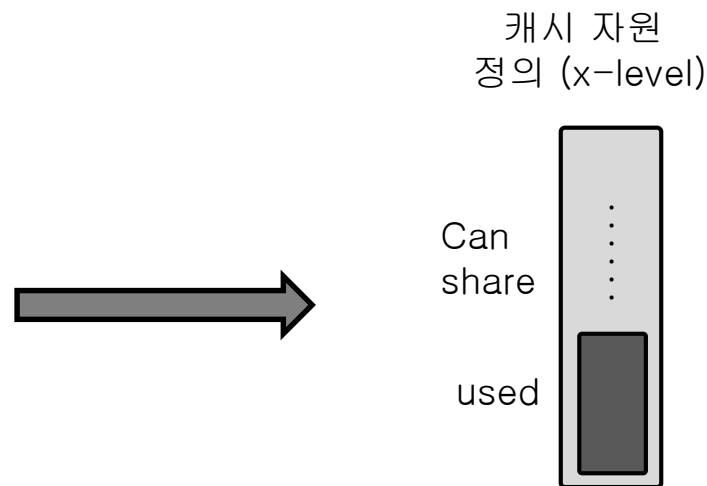
- 캐시 서비스 유형
 - 데이터 저장
 - Cache Utilization (CU)
 - 실시간 추론 서비스 데이터 저장
 - 서비스 체인 기반 응용 지원 기술
 - 학습 데이터 저장
 - 연합 학습 시 캐시 고려 사항 검토
- 일반적 캐시 할당 정책
 - Least Load & Rule : 자원 사용률이 가장 낮은 곳 추천, 수직 지원은 모든 캐시의 사용률이 85% 이상이며 클라우드 캐시 사용
 - DRL : 심층강화학습 기반 정책 생성

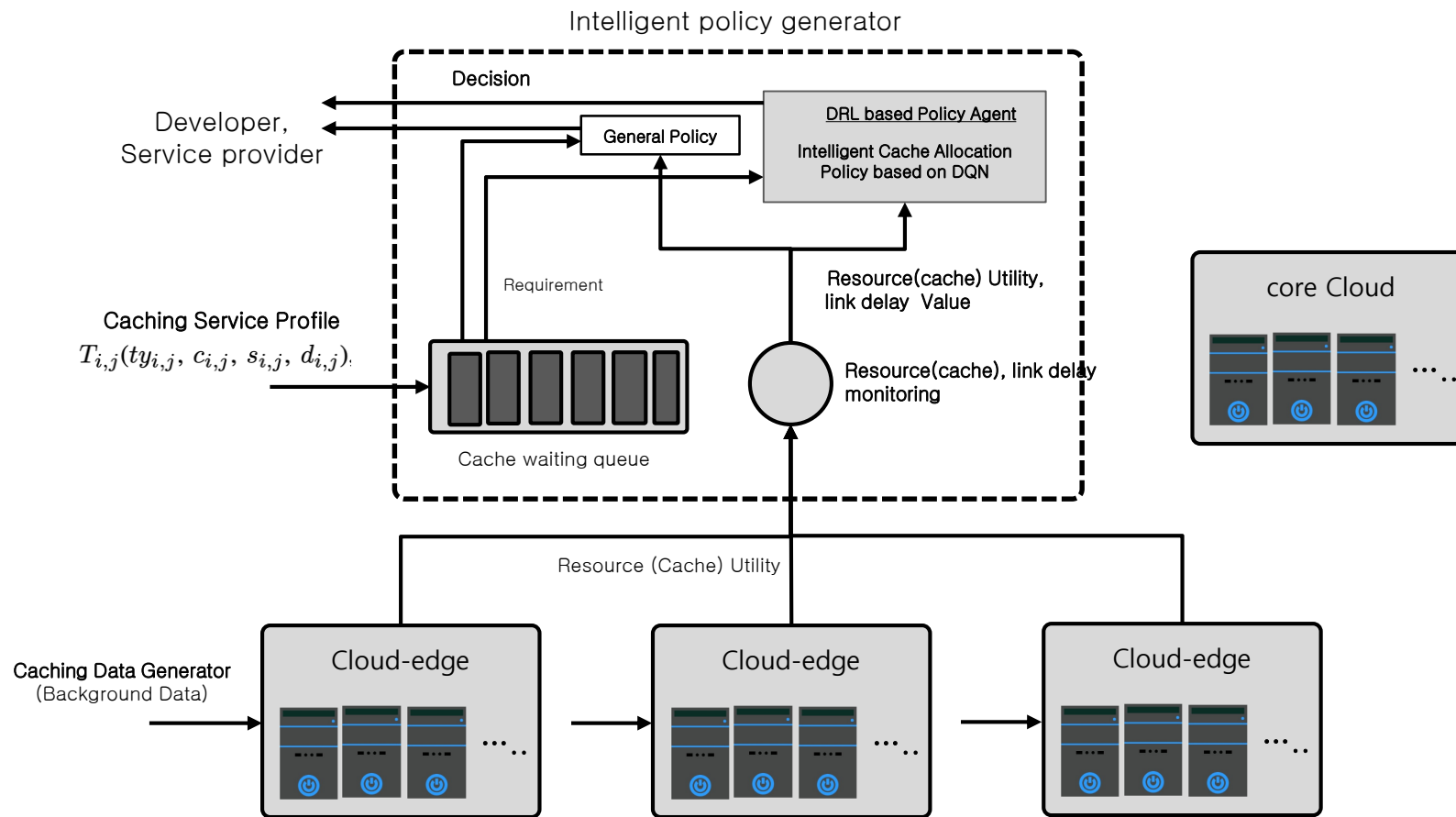
- 캐시 정책 목적
 - 캐시 네트워크 내 전체 캐시 사용률(CU) 최대화
 - DRL 정책 생성



5 DRL 기반 캐시 정책 모델

- 목적 함수
 - 전체 캐시 사용률 최대
 - $= \text{전체 사용 데이터 양} / \text{전체 캐시 크기}$
- State function
 - 캐시 사용률 기반 상태 정의
- Action function
 - 캐시 위치 지정 (수직, 수평 협업 포함)
- Reward function
 - 목적 함수내 최적 값을 얻기 위한 함수로 정의





IV

향후 연구



- 강화학습 기반 오프로딩 정책 생성기 학습 모델 개발
- 강화학습 기반 서비스 이동 학습 모델 개발
- 캐싱 정책 알고리즘 개발
 - MDP 모델 정의

감사합니다.

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



GS-Link 프레임워크 코어 개발자 (GS-Linkhq)
윤주상 (joosang.youn@gmail.com)

Welcome to GEdge Platform

An Open Cloud Edge SW Platform to enable Intelligent Edge Service

GEdge Platform will lead Cloud-Edge Collaboration