DEVIEW
2019

네이버 동영상 서비스를 지탱하는 VOD 플랫폼 개발기

조성철
동영상플랫폼
NAVER

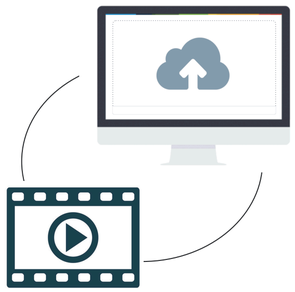
CONTENTS

DEVIEW
2019

1. VOD 서비스에서 고려 사항

VOD 서비스 Flow

INGEST



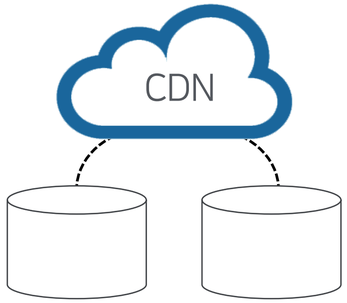
다양한 기기에서 캡쳐
동영상 업로드

CONVERT

Transcoding
Transmuxing

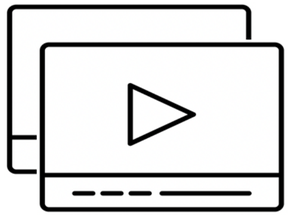
코덱, 컨테이너 변환
HLS, DASH

STORE & DELIVER



Streaming, AD,
DRM License

PLAYBACK



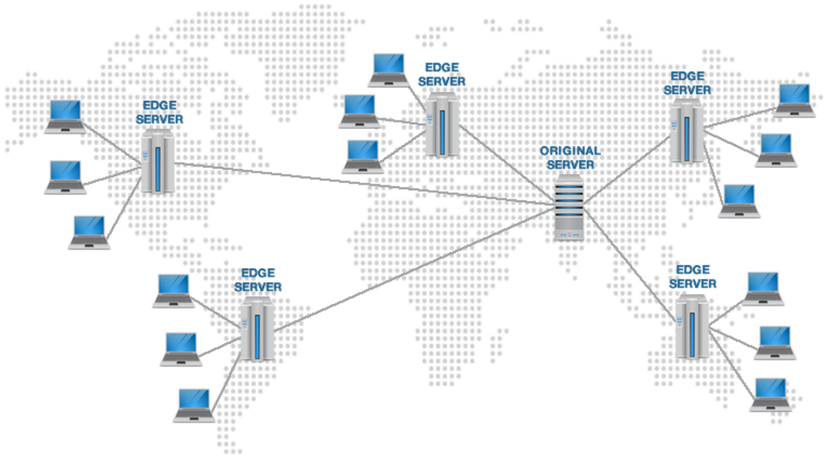
다양한 환경에서
재생

ANALYZE

품질, 소비 분석

다양한 환경에서 동영상 재생

파편화된 동영상 재생 환경
데이터 압축과 컨테이너 변환
글로벌 서비스를 위한 딜리버리

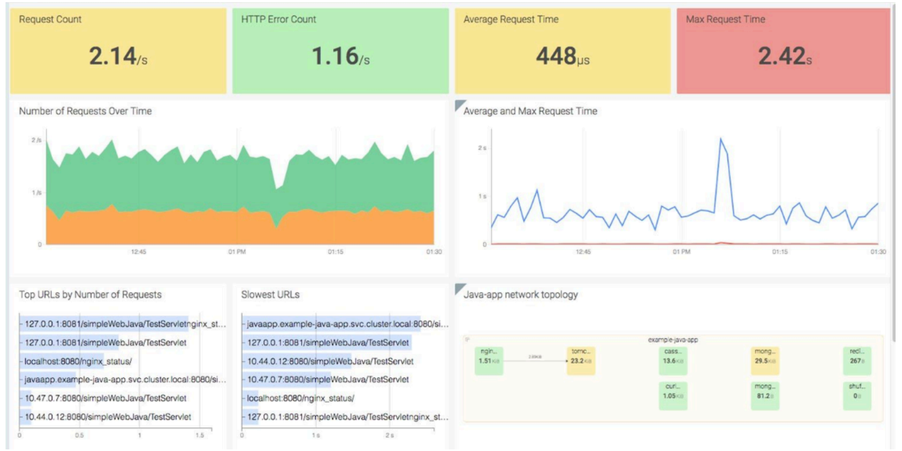


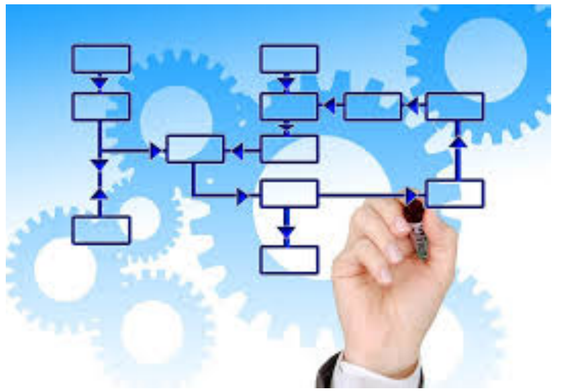
대용량 시스템의 운영

대용량 파일 업로드와 데이터 전송

Object Storage 저장 및 관리

배포, 모니터링, 장애 대응 등





수익화 및 분석 도구의 개발

AVOD, SVOD?

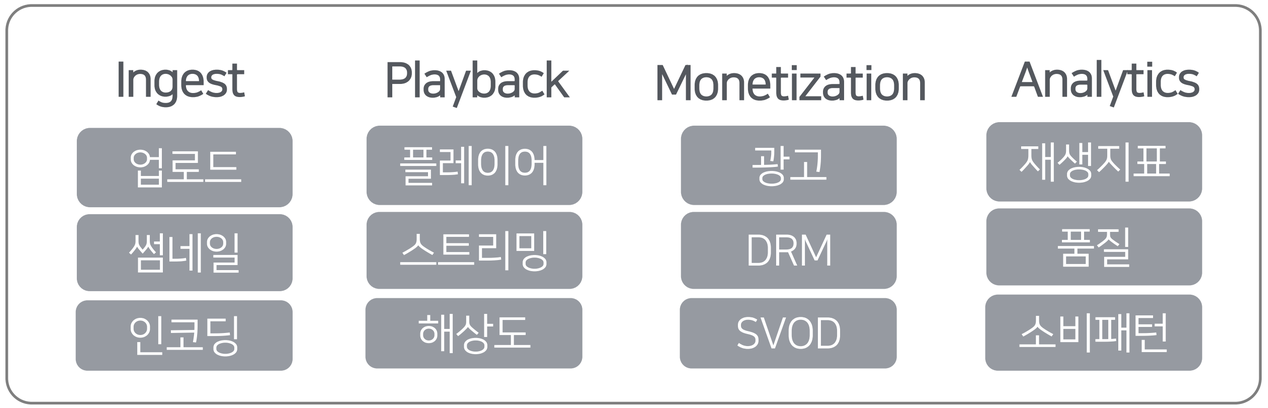
동영상 광고 연동 (VAST, SSAI, DFP)

동영상 콘텐츠 보호(DRM)

소비 분석 도구, Analytics

Today

VOD 플랫폼을 개발하면서 겪은 **경험** 공유



DEVIEW
2019

2. 업로드부터 트랜스코딩까지 최적화하기

동영상 대형화로 나타나는 문제

DEVIEW 2019

기존에는 다 잘 되던것들 인데…

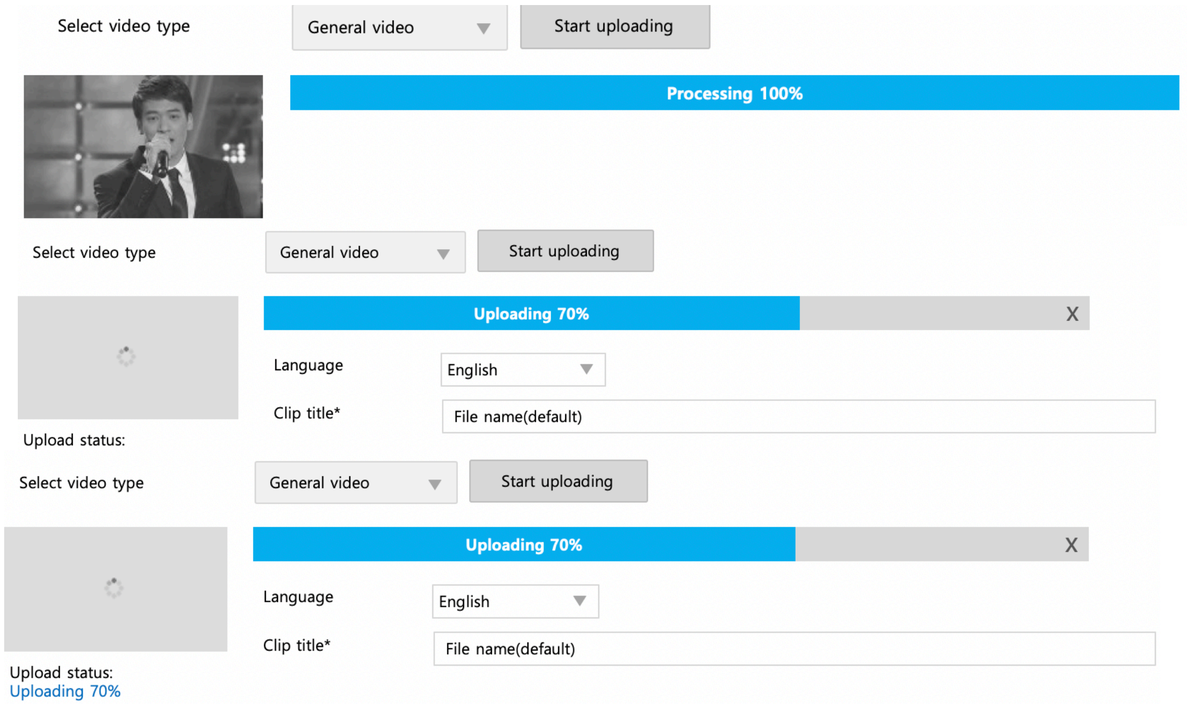
업로드 오류 증가, 시스템 트래픽 부담, 트랜스코딩 지연

업로드 방식의 변화

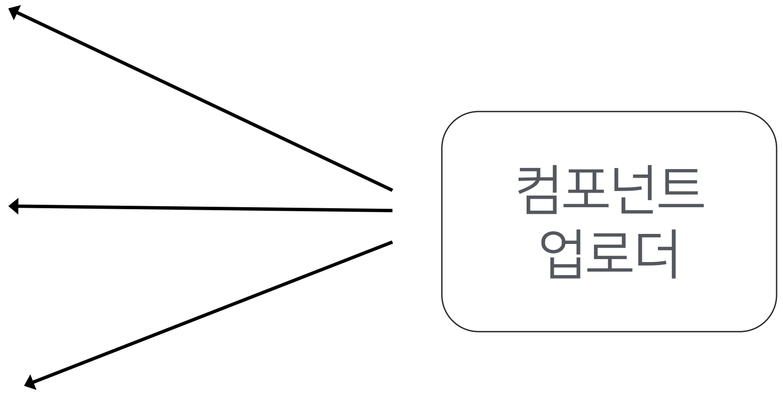
DEVIEW 2019

한번에 여러개의 파일을 업로드 하고 싶어요.

멀티 파일 업로드, 업로더 UI도 서비스에서 직접 구현했으면…



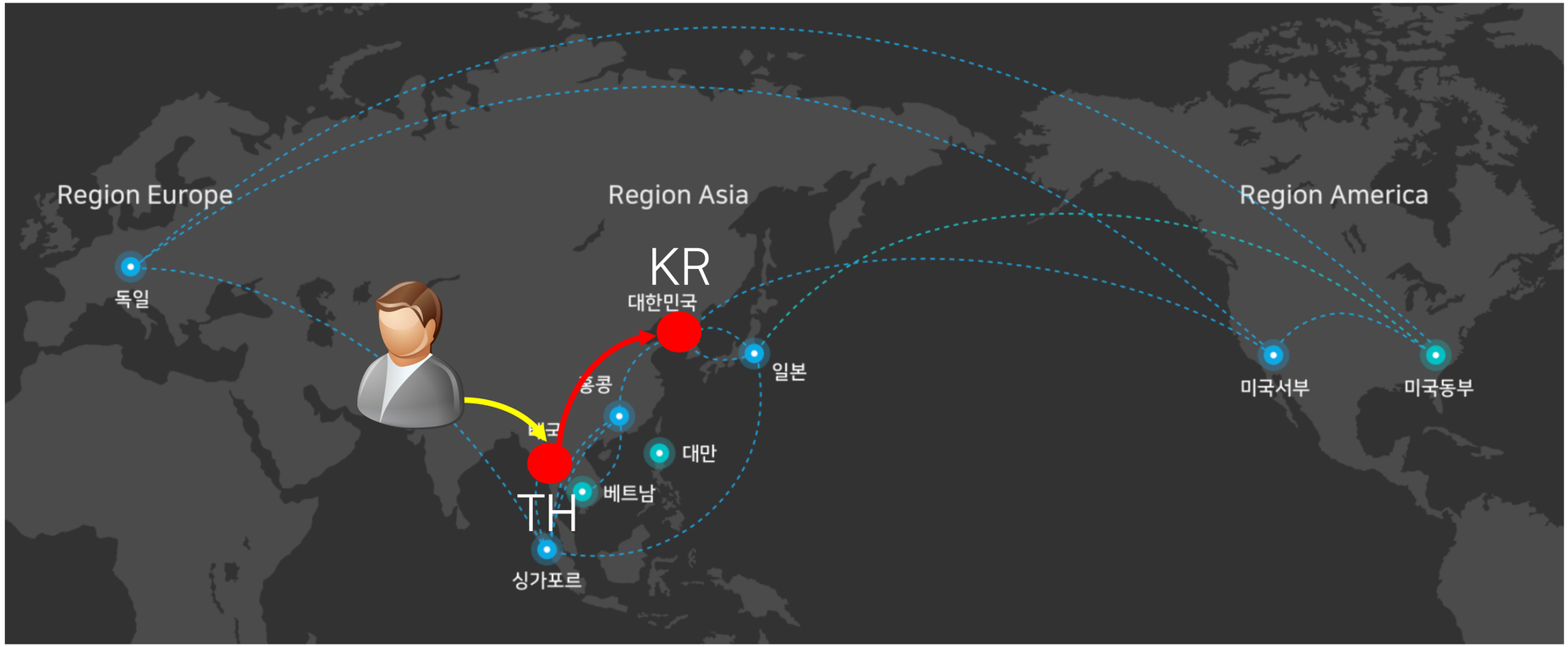
…



특히,해외에서 업로드가 이슈

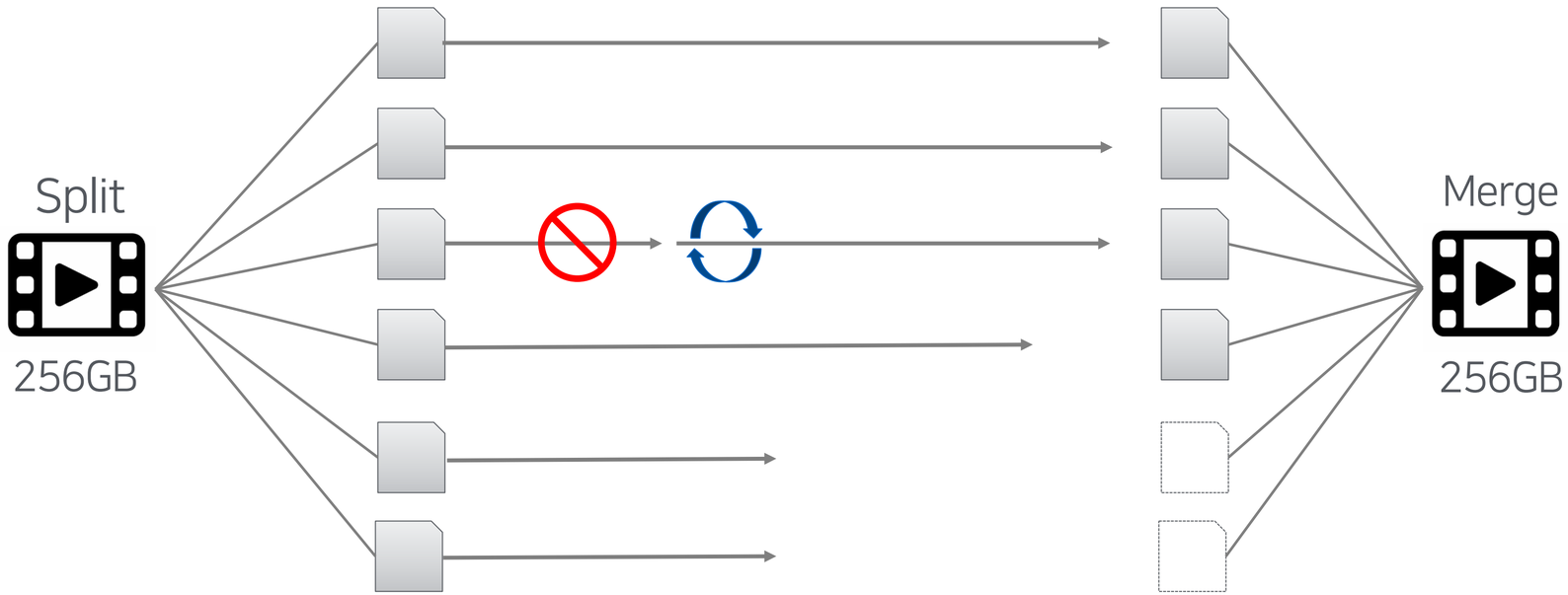
해외 CP사에서 대용량 파일을 받기 힘든 상황

KR POP에만 VOD플랫폼이 구축, 해외 POP에서 국내로 재 전송하는 구조



업로드 모듈의 기본 컨셉

대용량 파일 전송, 실패 자동 복구, 멀티파일 업로드
개발, 배포가 쉬운 컴포넌트 구조

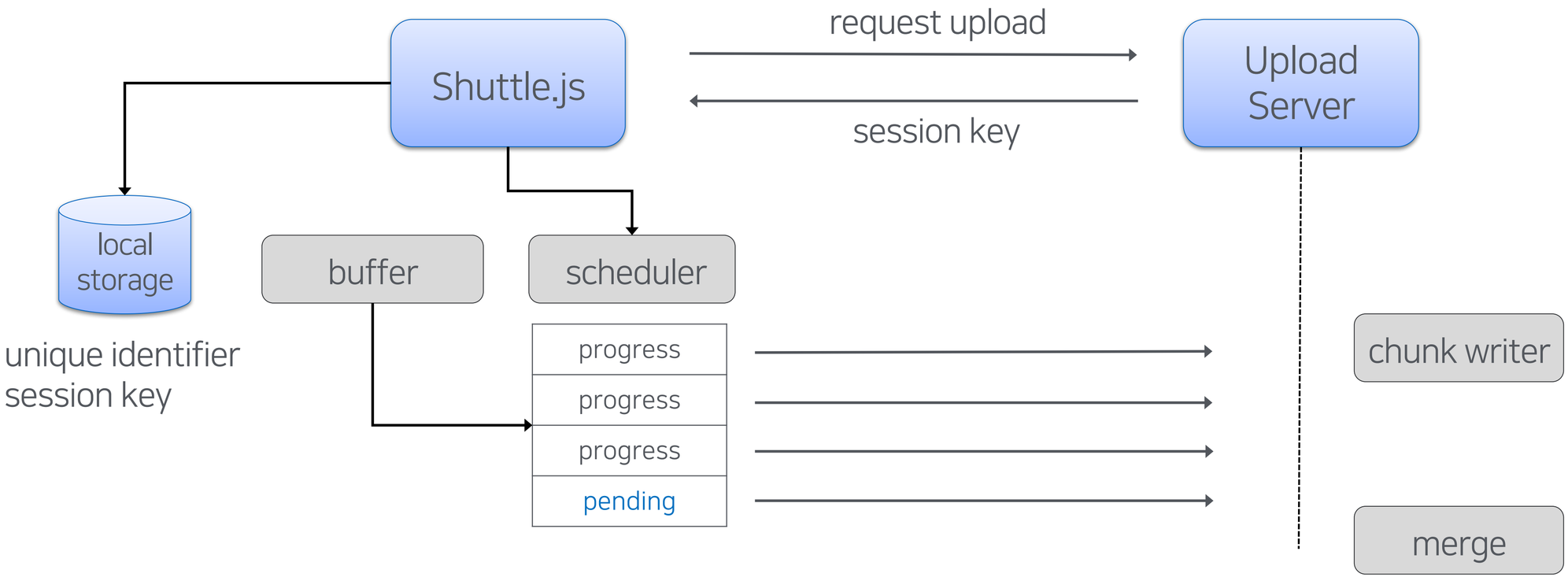


Chunk 단위로 파일을 전송하는 구조

DEVIEW 2019

identifier + session key로 저장

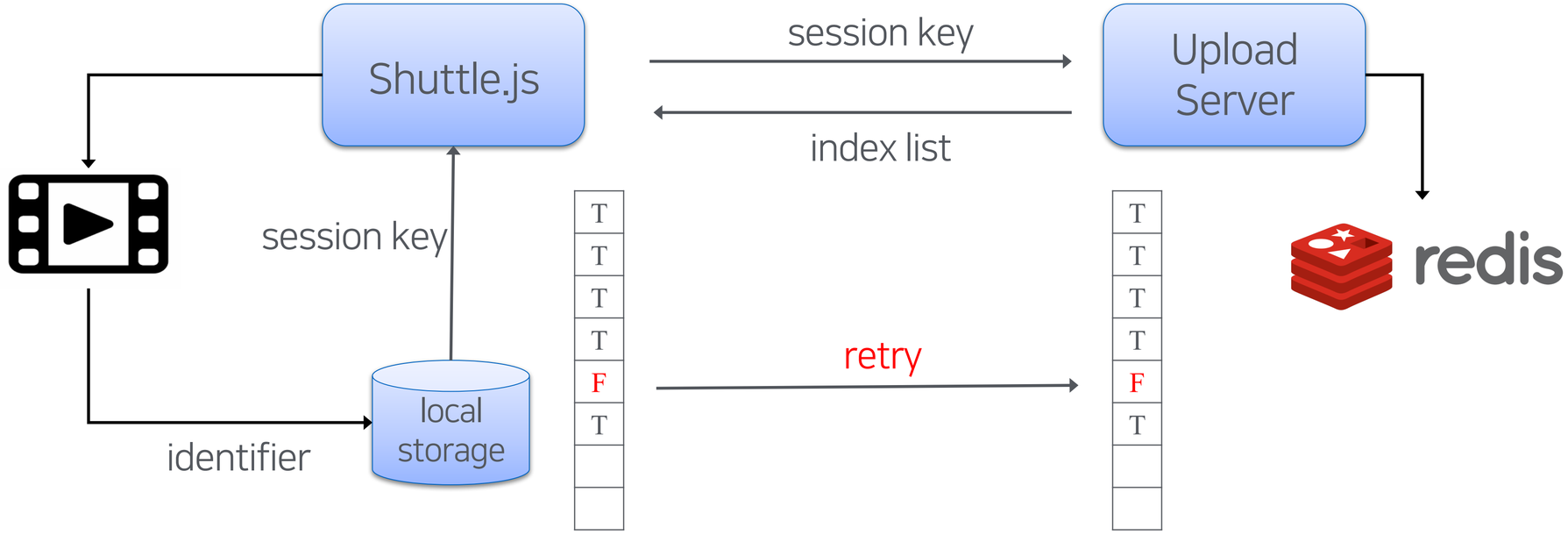
buffer에서 chunk 생성, scheduler 통한 전송 구조



Auto-Resume의 동작 방식

DEVIEW 2019

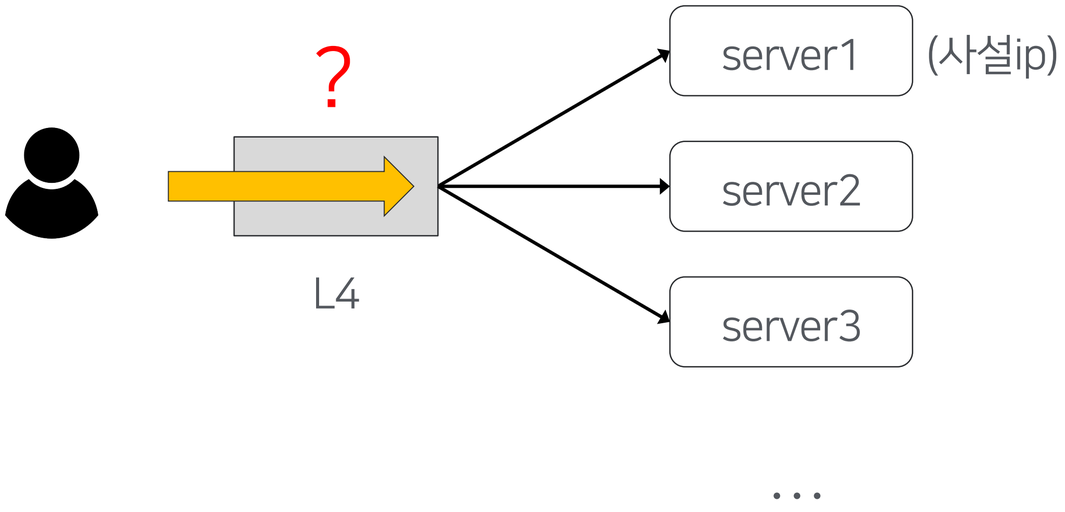
Client, Server 간 전송 list 동기화, 실패한 Chunk를 전송

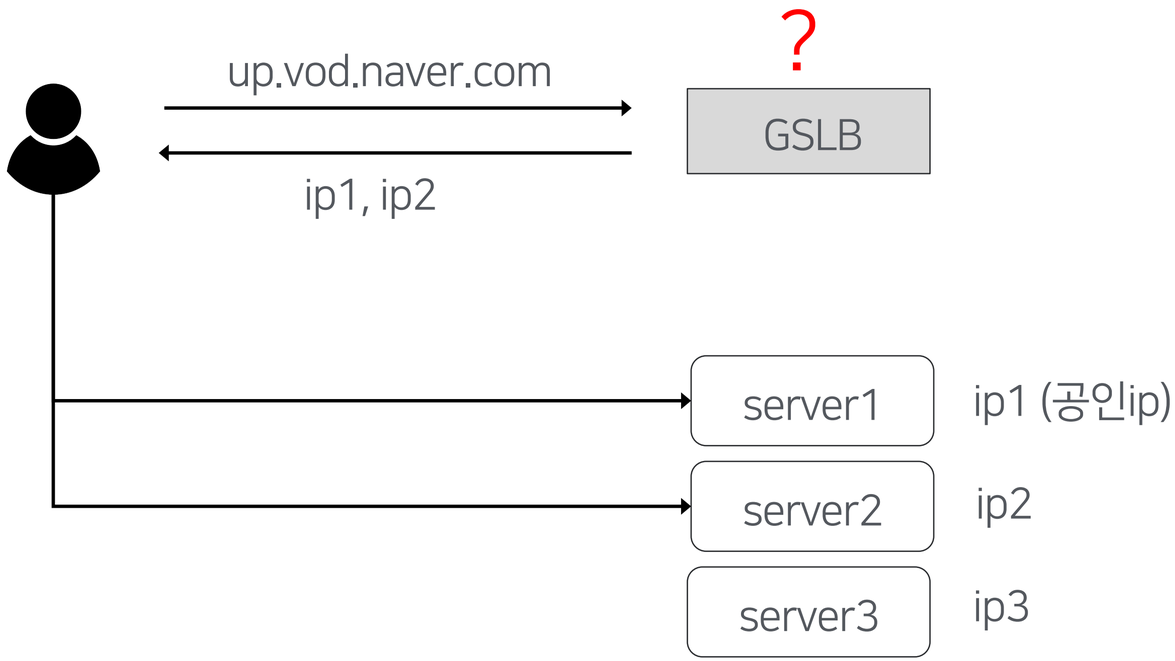


업로드 서버의 Load Balancing

L4, 대용량 트래픽 전송 문제

GSLB, DNS 캐싱으로 빠른 갱신이 안되는 문제

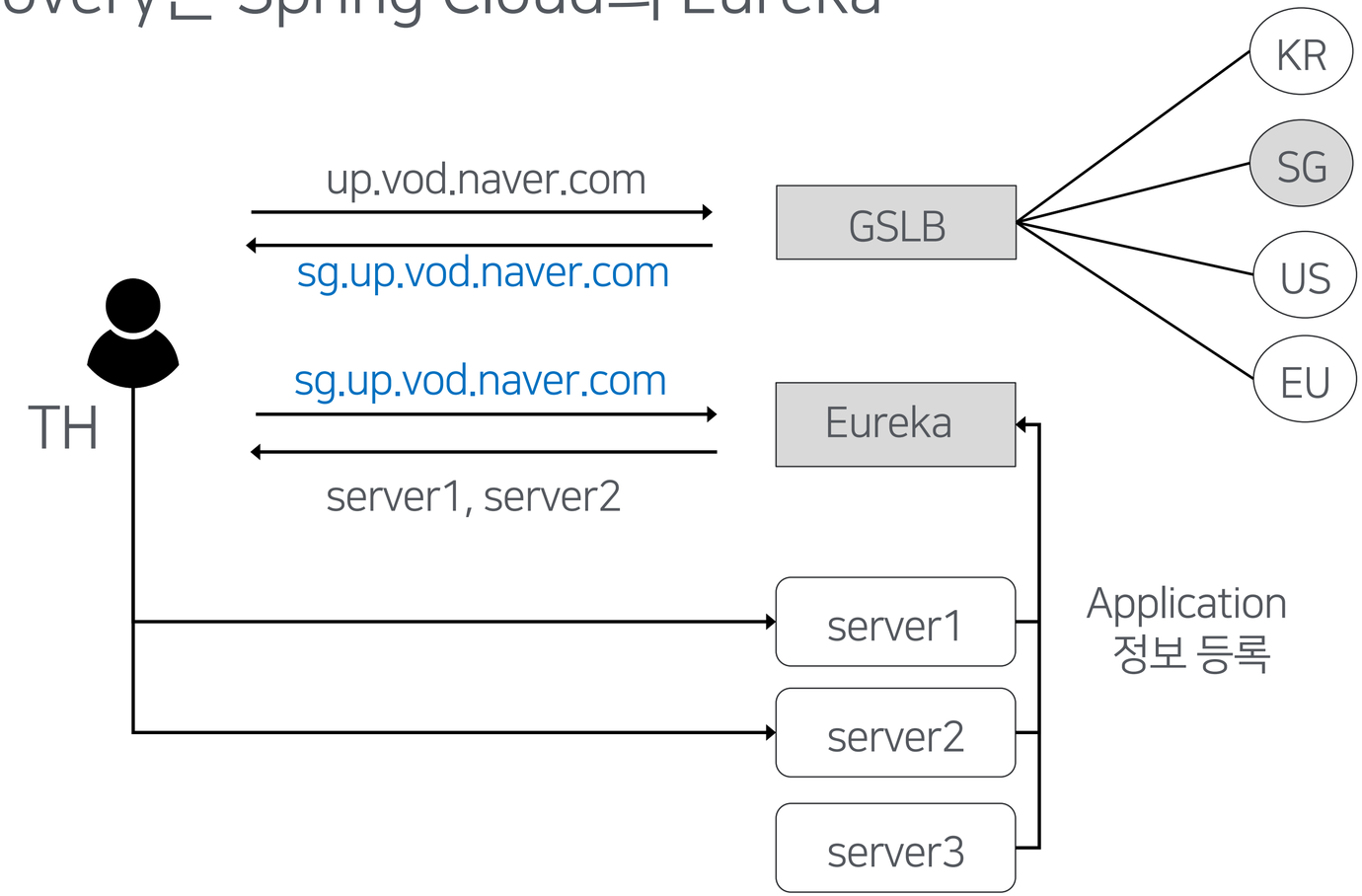




GSLB + Eureka를 이용한 방법

GSLB Geolocation으로 가까운 POP 할당

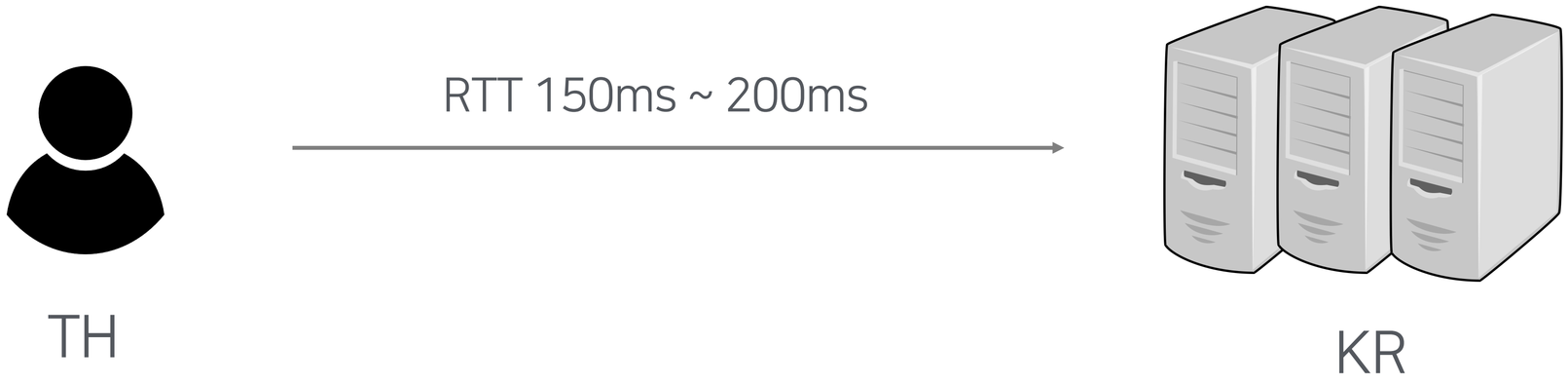
Service Discovery는 Spring Cloud의 Eureka



해외 구간의 전송 속도 개선

불안정한 로컬 네트워크 환경 (packet loss)

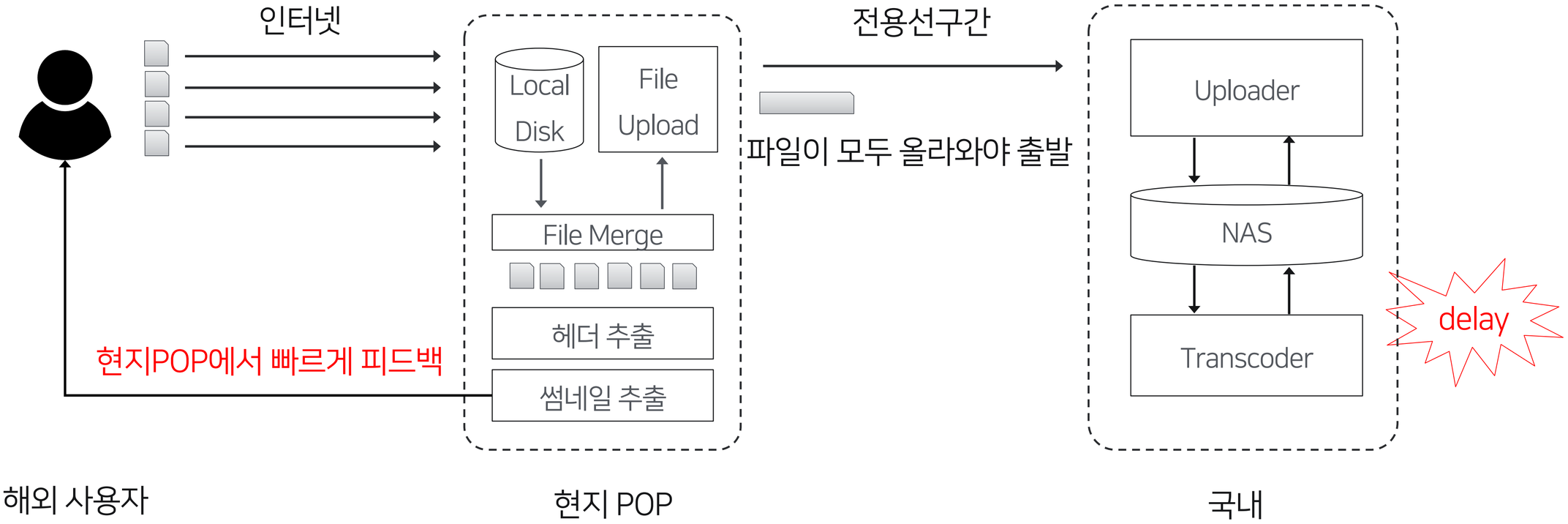
국내 IDC까지 전송해야 함, 퍼블릭 인터넷만으로는 업로드 불가능



Global 업로더 구조 개선 #1

파일이 커지면 재전송을 위한 대기 시간이 길어짐

사용자에게 업로드 후 바로 피드백이 가능

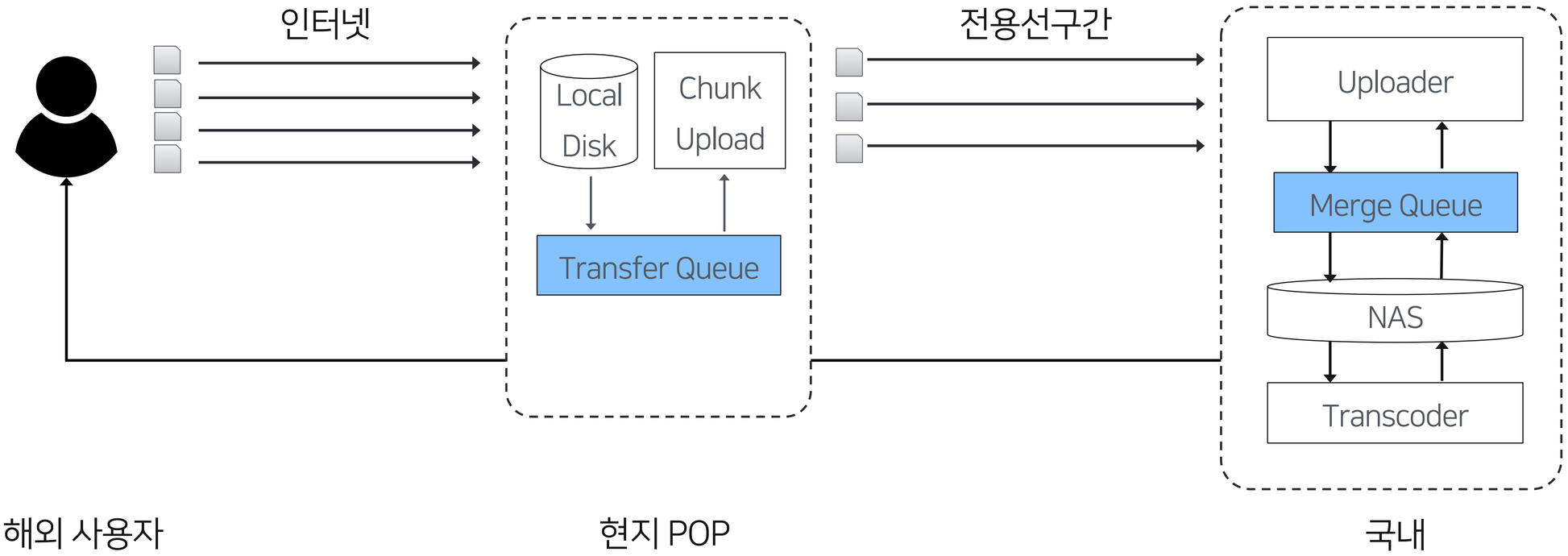


Global 업로더 구조 개선 #2

Transfer Queue 를 통해서 Relay

File Merge 는 국내 서버에서만 처리

전송 지연이 거의 발생하지 않음

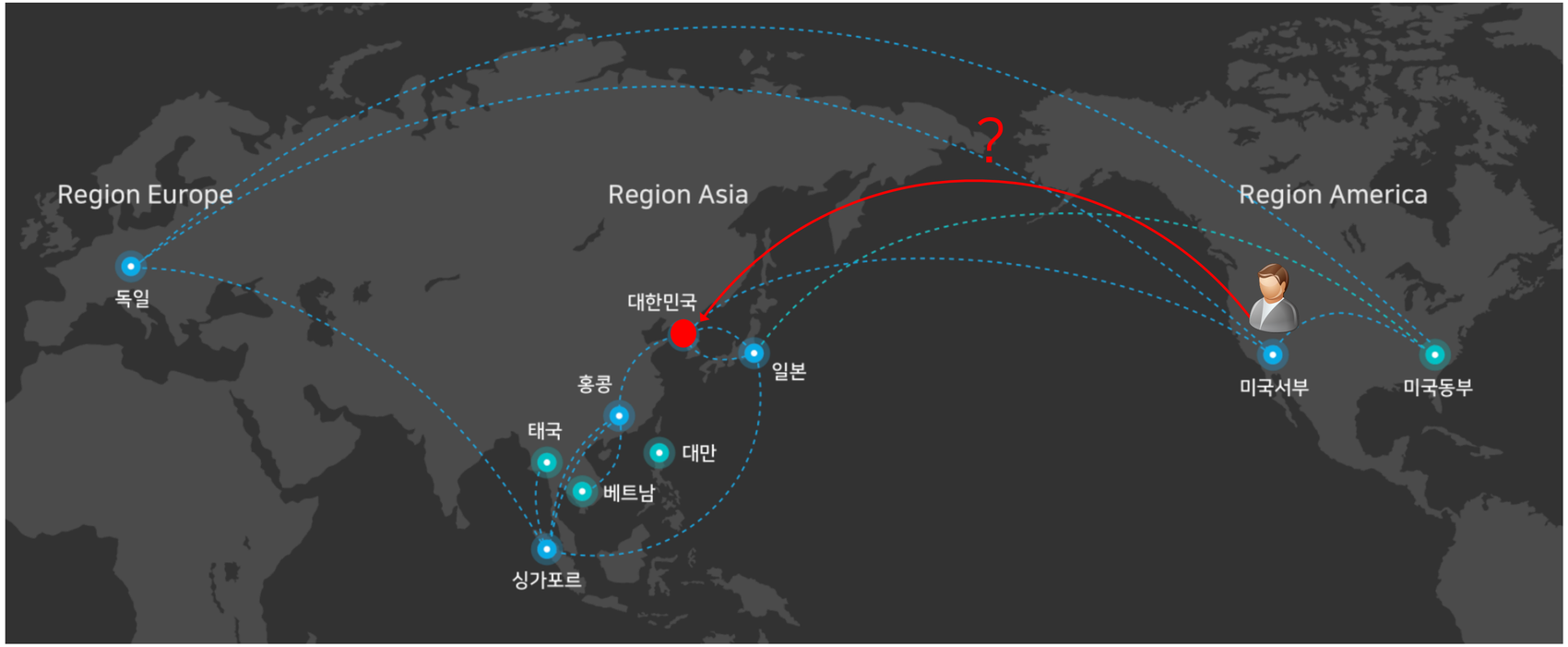


Global 업로드 모델의 한계

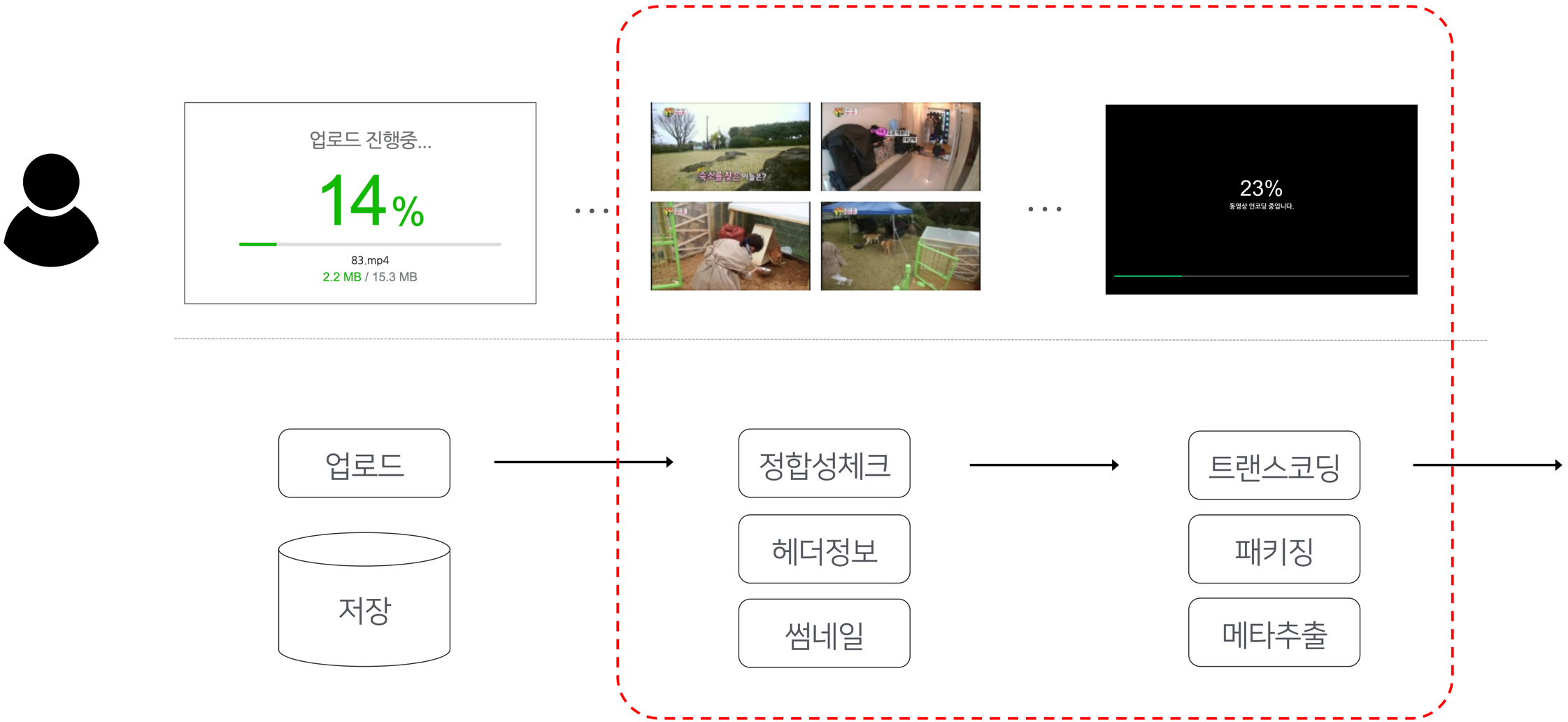
동영상의 대용량화는 가속

전용회선 B/W의 제약, 글로벌 POP의 확장 지원

트랜스코딩 클러스터, 오브젝트 스토리지, 분산 DB 구축



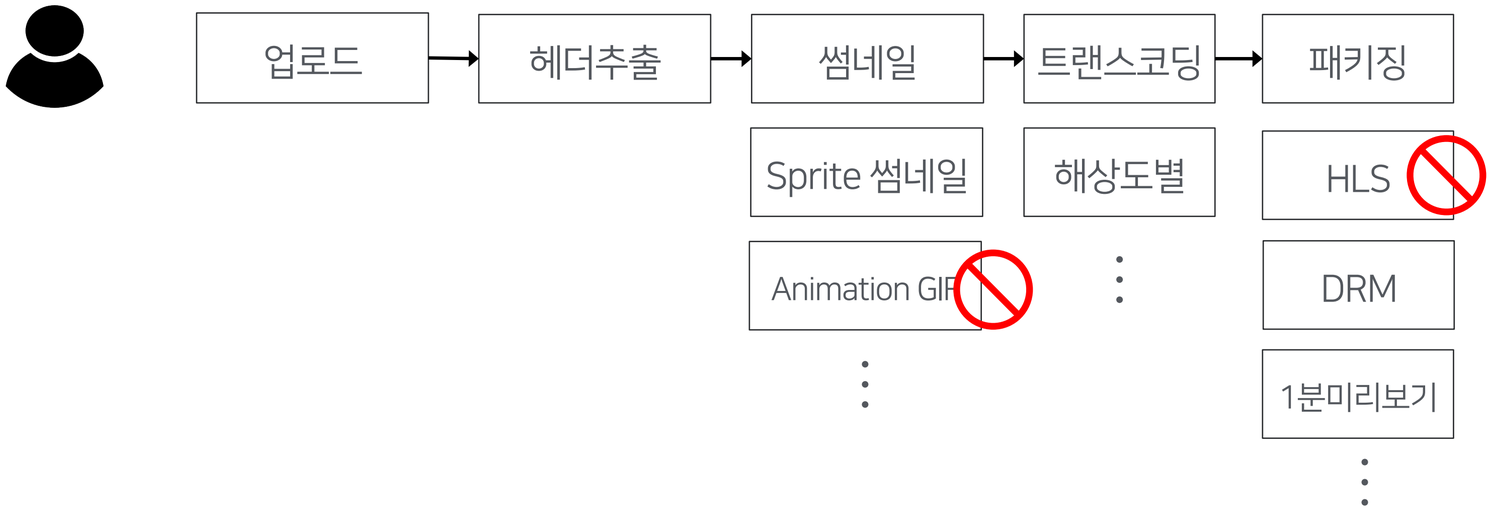
업로드이후 작업 프로세스



Ingest 단계의 작업들을 관리하자

10단계 이상의 Task가 존재, 계속 늘어나는 구조

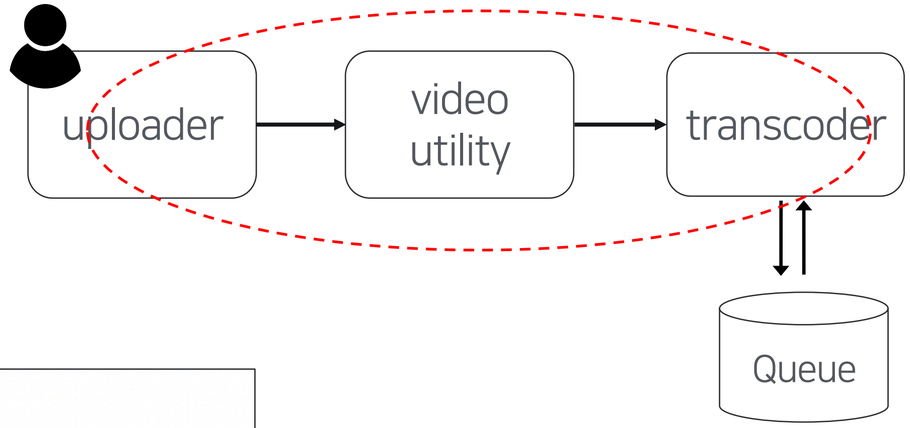
Task에 대한 제어, 실패시 자동 복구가 필요

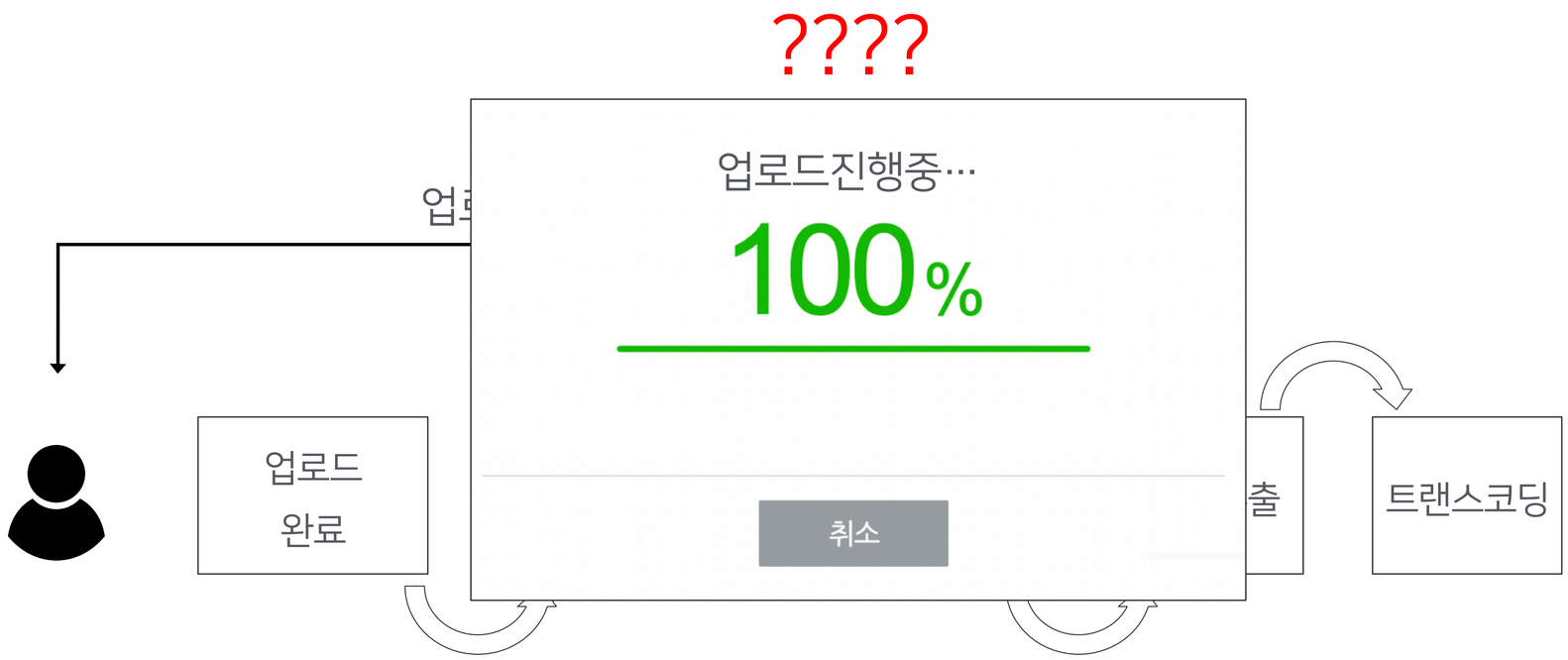


기존 구조의 문제점

단계가 연결된 시스템 구조

단위 작업 실패시 복구가 어려움

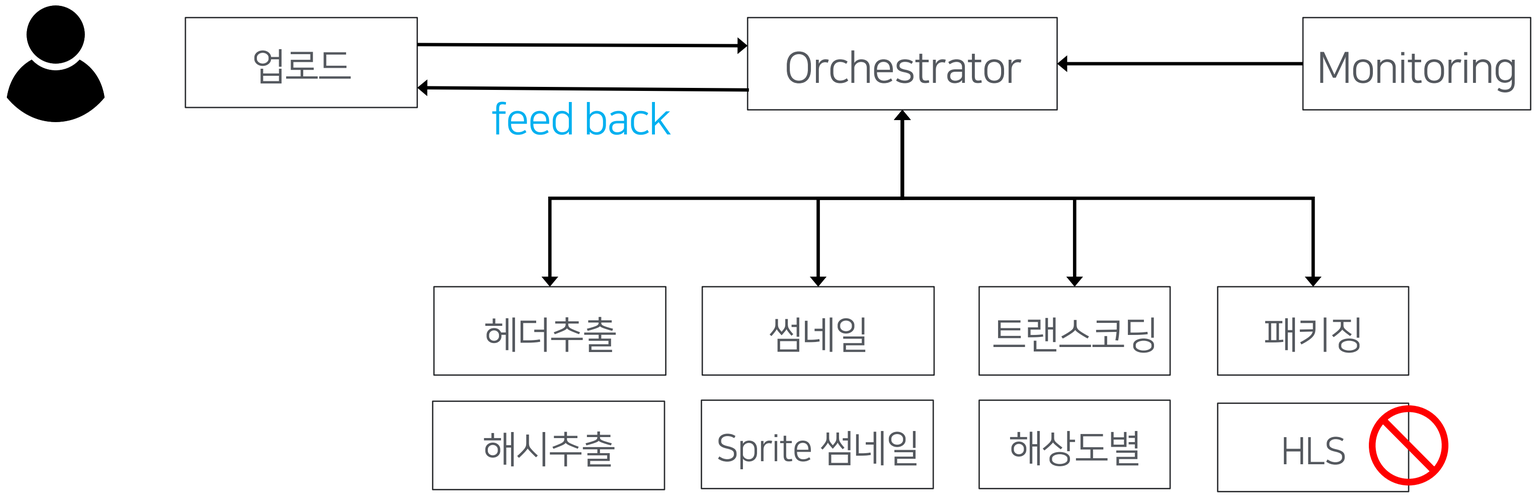




Orchestrator를 만들고 위임한다.

Task간 Dependency 제거

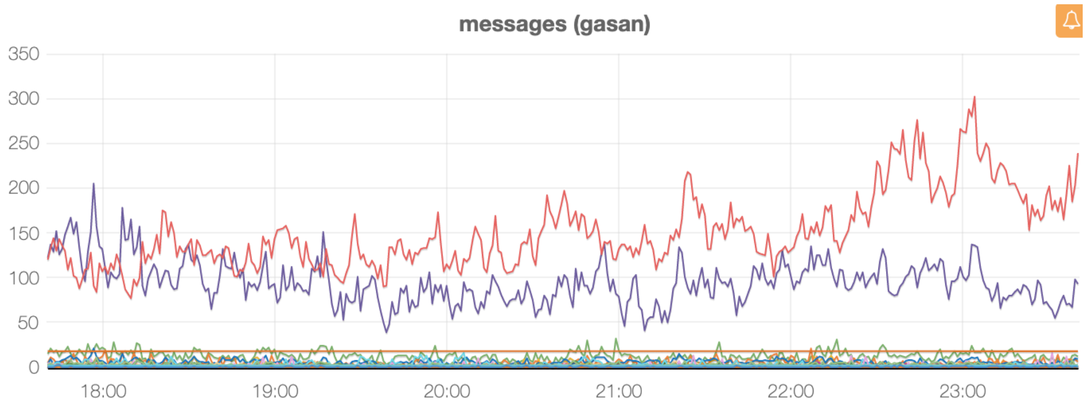
이제 모든 단계의 Feed Back이 가능해졌다

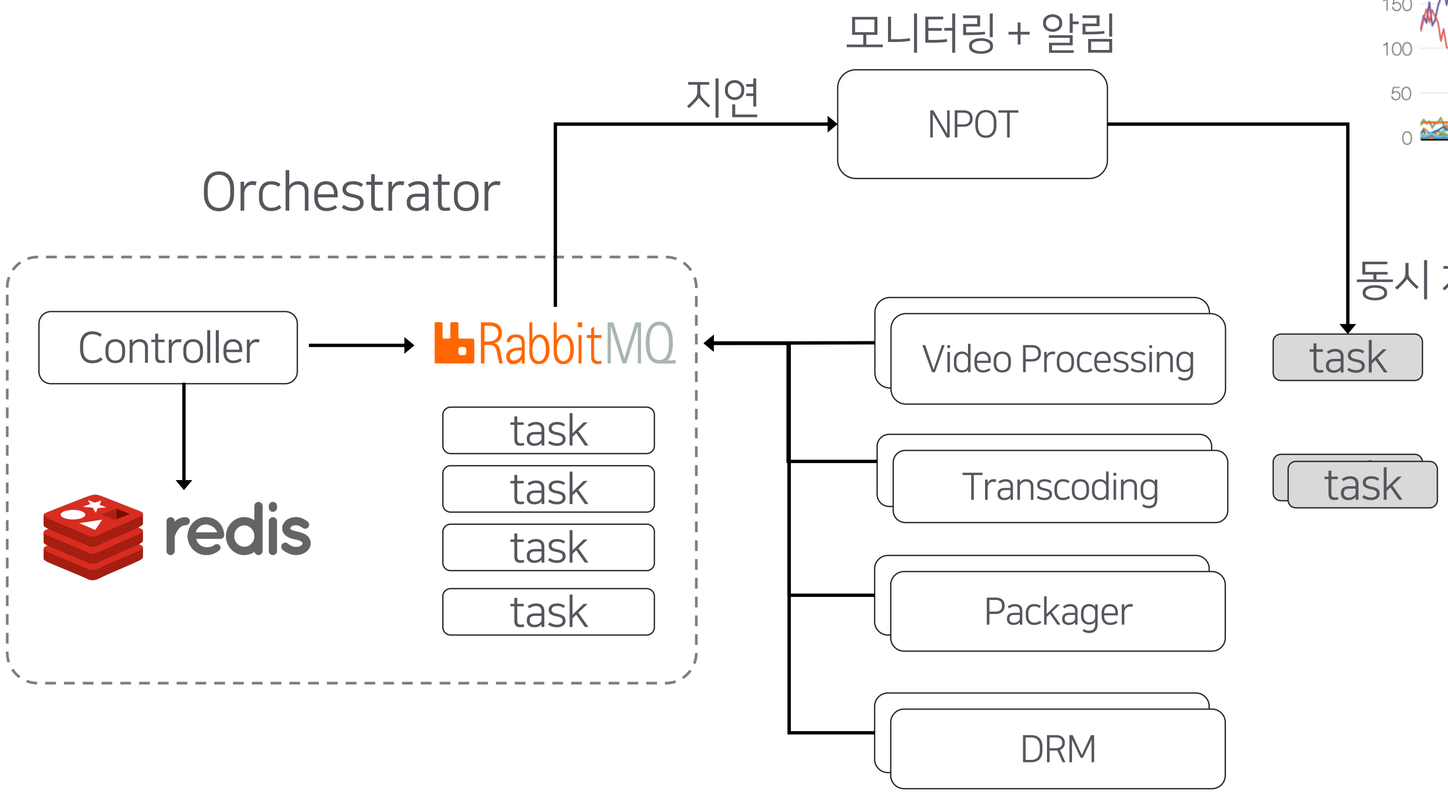


작업 제어와 모니터링을 좀 더 편리하게

작업별로 리소스 사용량이 다름

Task 요청 증가, 지연시 Consumer 조정

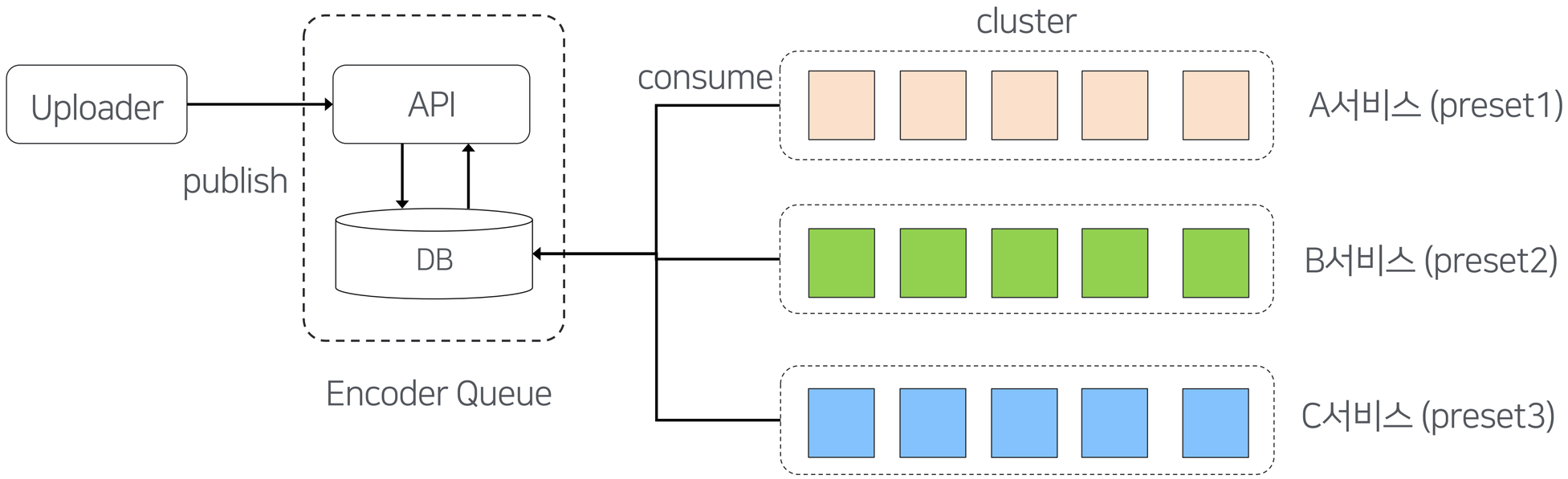




기존 트랜스코딩 클러스터 상황은?

encode item을 DB에서 관리, preset을 미리 만들어서 주입하는 방식

서비스별 인코딩 설정이 다르면 별도 클러스터로 만들어야 되는 문제

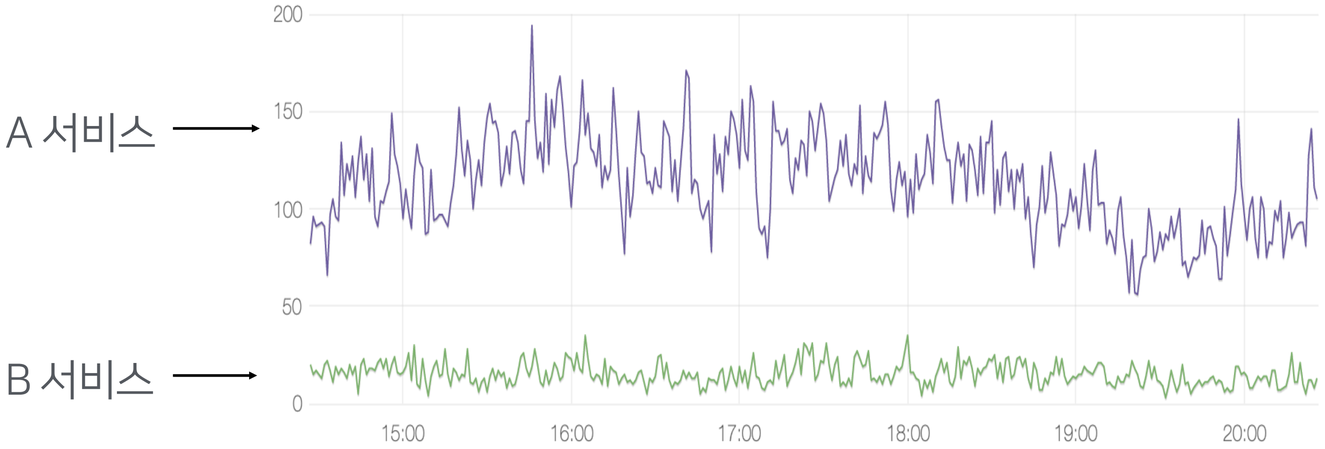


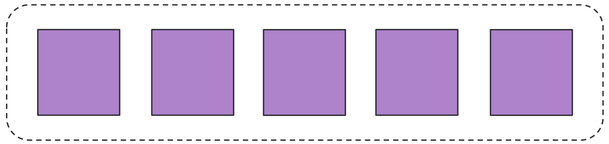
로드밸런싱 문제

DEVIEW 2019

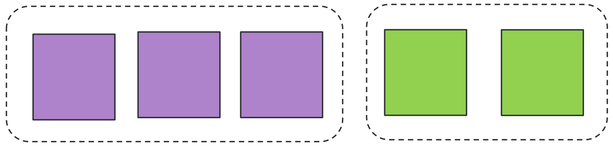
서비스별로, 시간별로 요청량이 다름

트래픽 증가시 수동으로 팜을 늘려줘야 하는 구조



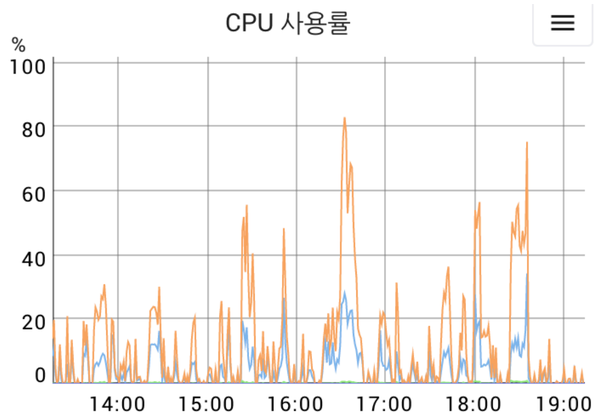


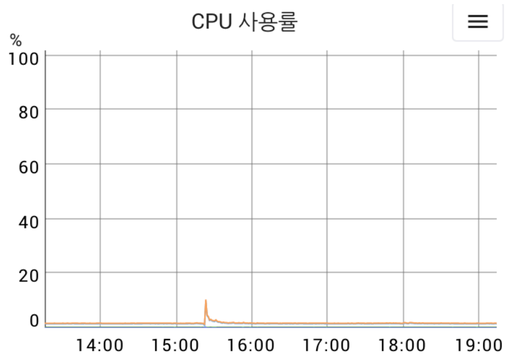
A서비스



B서비스

A서비스로 투입



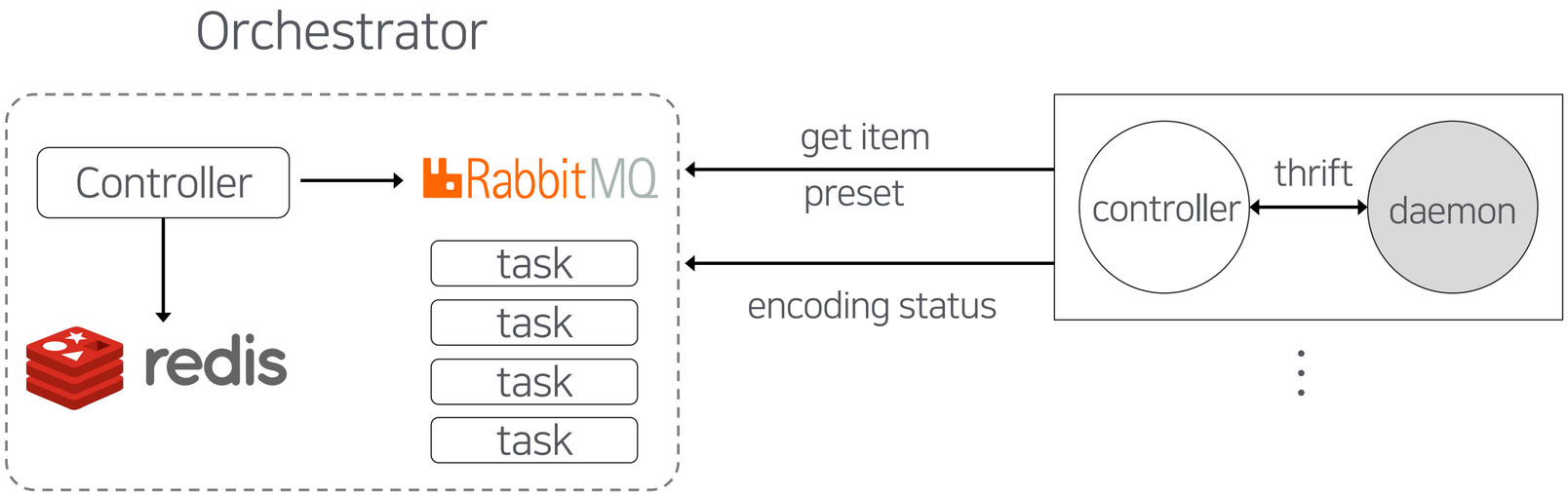


트랜스코더 클러스터 구조 개선

자체 트랜스코더로 내재화, Preset를 Run-time에 주입

Orchestrator에서 Task 할당, DB 의존성 제거

서비스별 분리된 Cluster를 통합 운영



새로운 트랜스코딩 시스템의 요구사항

저녁에 6시간짜리 Full Clip이 올라오면 새벽이나 되야 노출

단일 시스템의 속도 제약 해소

대용량 시스템의 배포, 운영, Scale-out

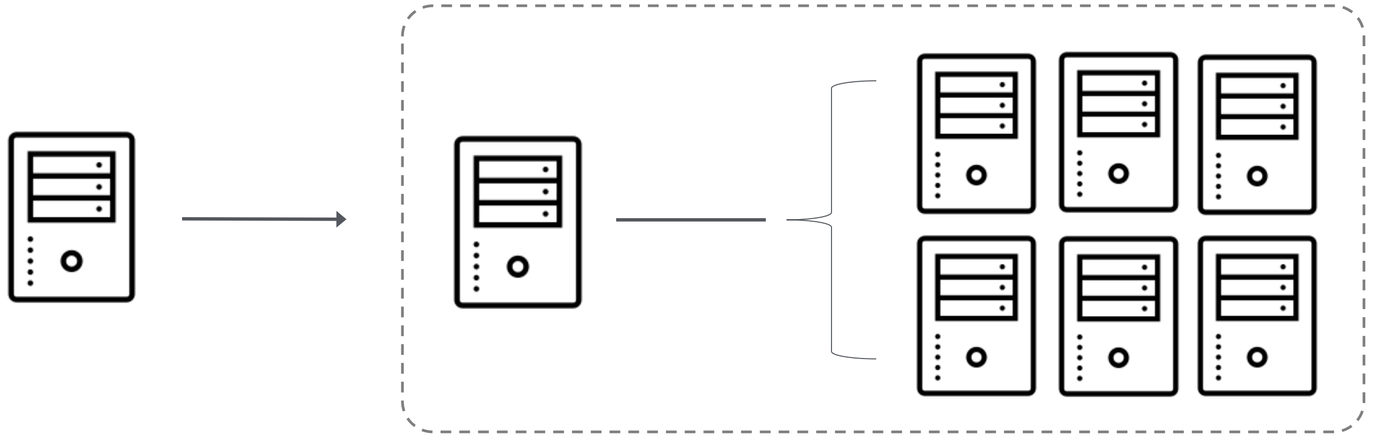
스마트한 리소스 밸런싱?

분산 트랜스코딩의 컨셉

원본 영상을 여러 조각으로 분할

다수의 장비에서 트랜스코딩

하나의 영상으로 병합



controller
(인코딩 job 관리)

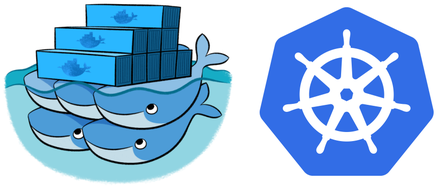
distributor
(분할/병합)

workers
(분할 영상 인코딩)

분산 시스템의 리소스 제어?

오픈 소스 기반, 컨테이너 기반의 쉬운 운영

사내 플랫폼 Docker Swarm



host 환경의 사용이 간편

resource scheduling

network support

Apache MESOS™

Distributed Kernel

Application에 적합한 Framework 사용

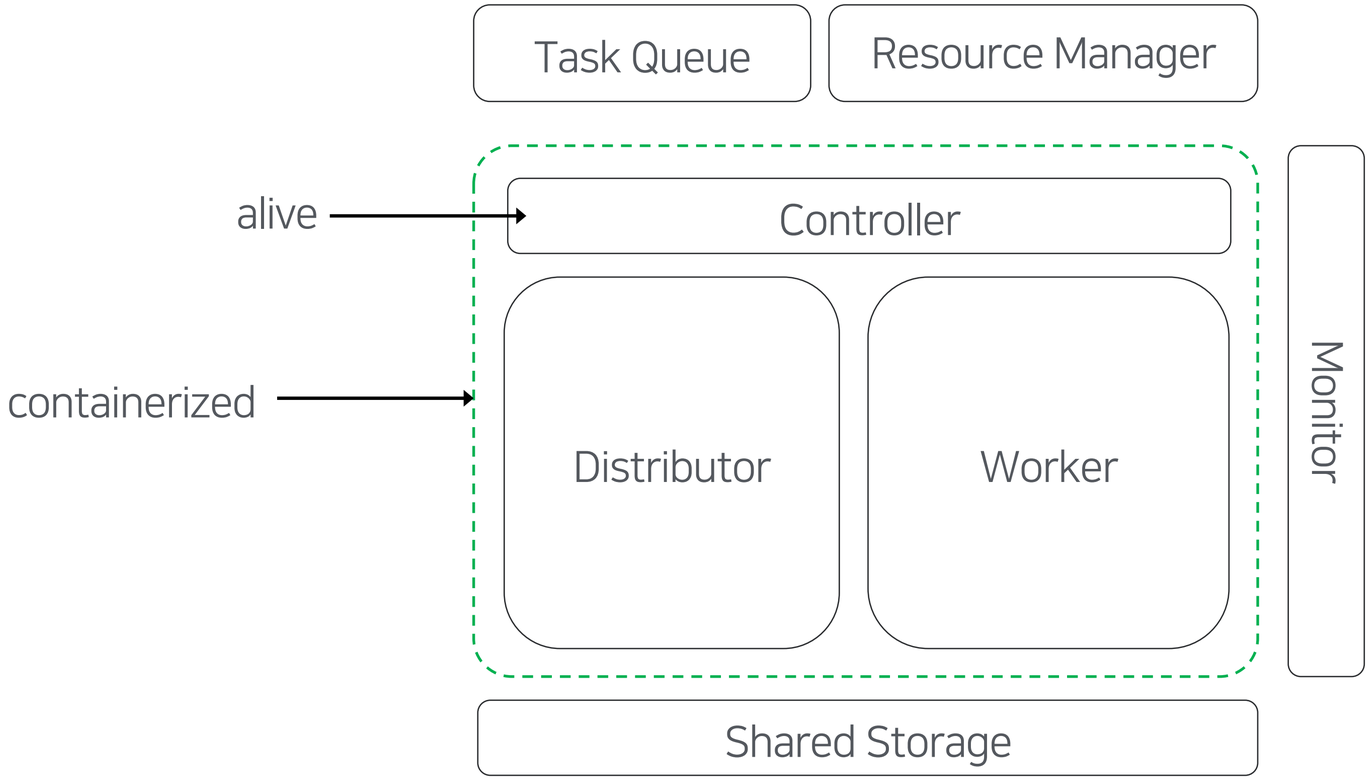
Hadoop echo system에 최적화

개발 초기 단계 프로토타입

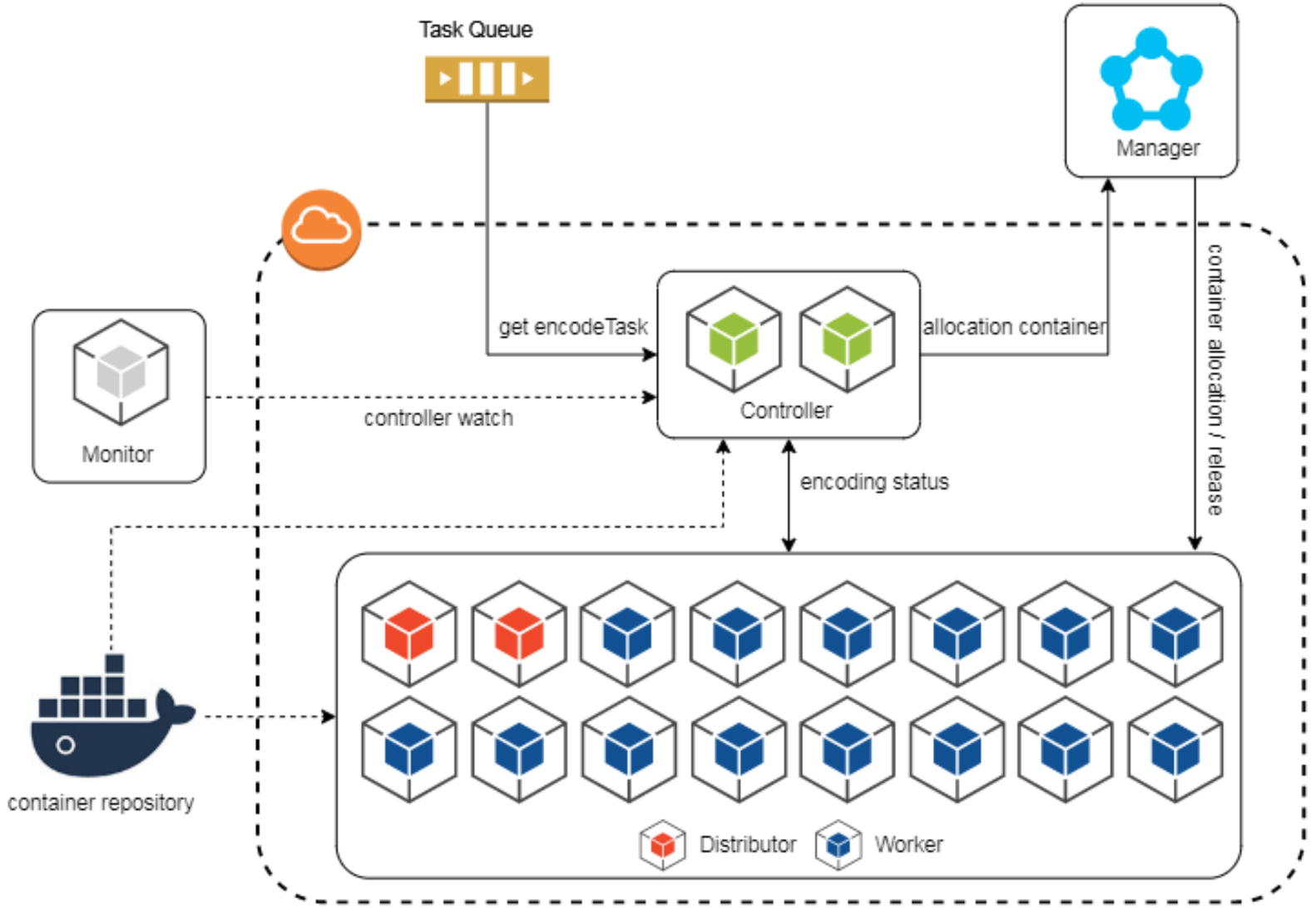
DEVIEW 2019

Containerized -> Controller, Distributor, Worker

Resource Manager -> Docker Swarm



분산 트랜스코딩 시스템 전체 구조

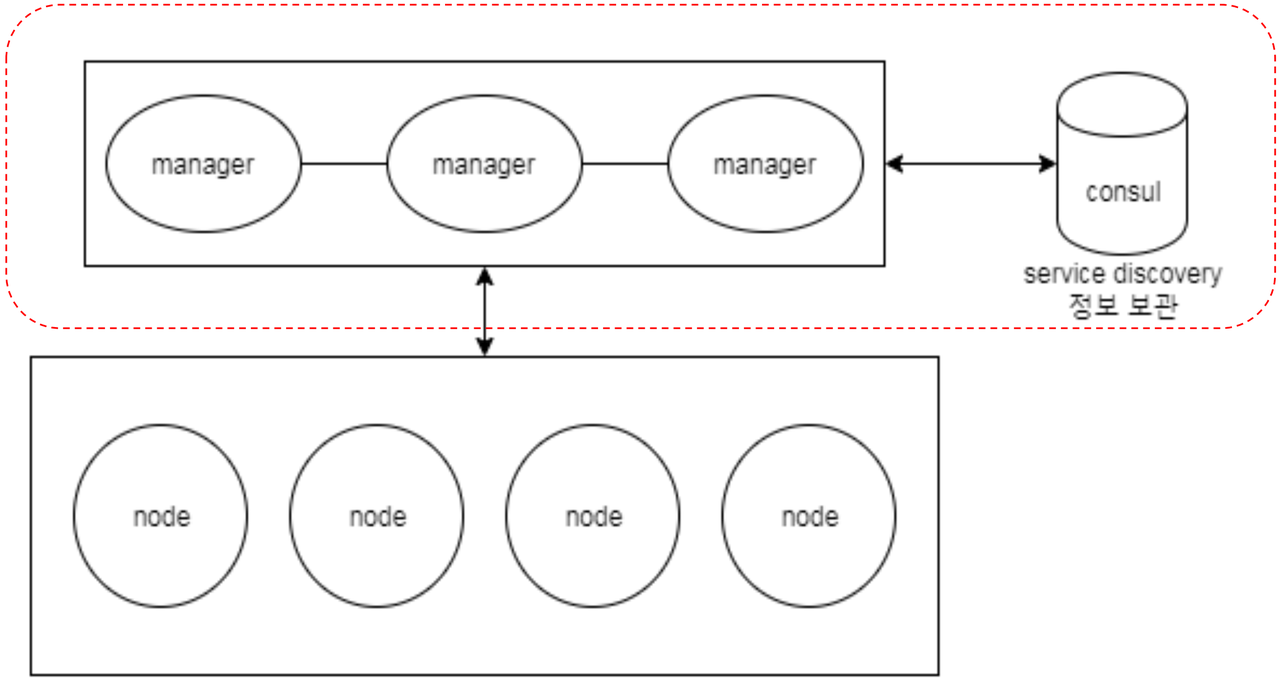


대량 컨테이너 Deploy시 지연 문제

분산 효과를 위해서는 많은 worker가 필요

200개 worker 동시 배치에 최대 20 ~ 30초 정도

20개 단위시 약 30ms 소요, 순차적인 deploy scale out



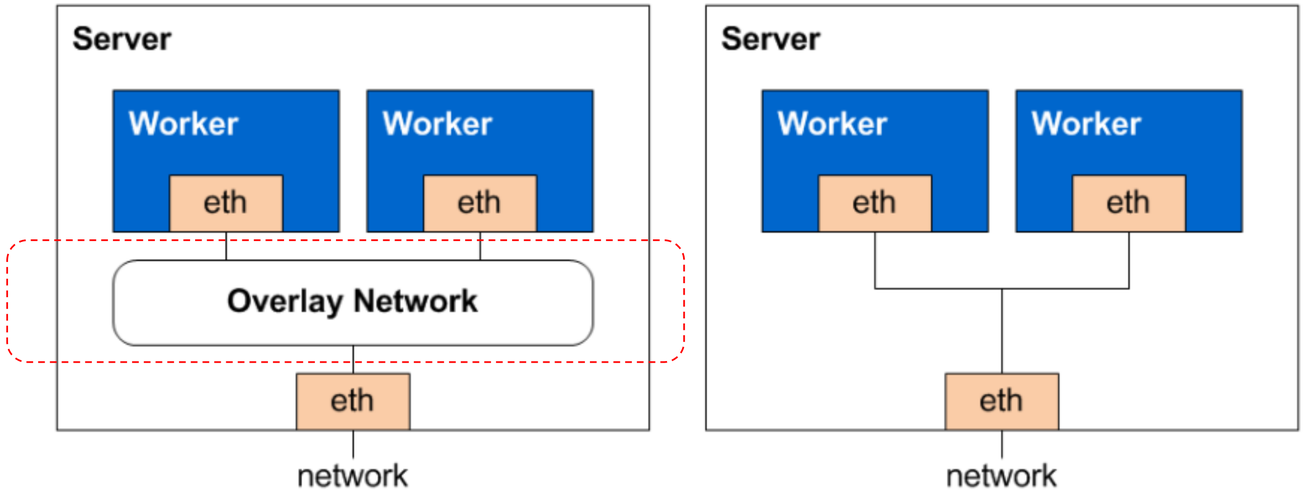
병목구간

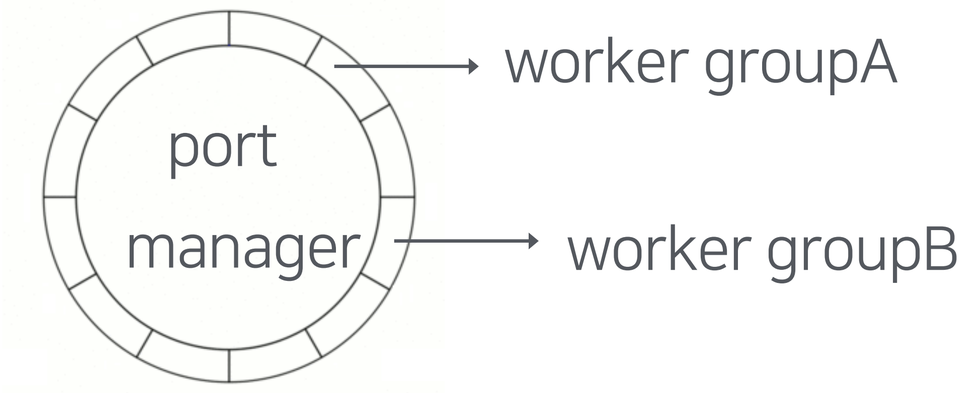
네트워크 속도 저하 문제

Worker들 간의 동영상 전송시 Overlay Network의 병목

Host Network 를 사용하도록 변경, 포트 관리가 문제..

컨테이너간 충돌 방지를 위해 Group 단위로 포트 분배



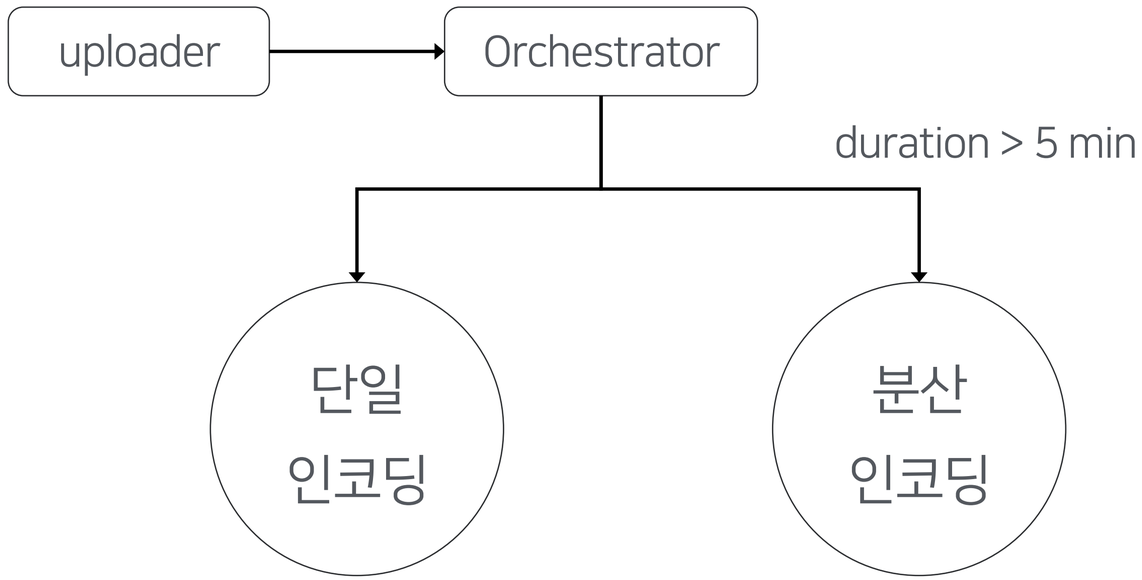


Ex) 20,000 ~ 21,000

분산 트랜스코더 적용은?

대량 Container Scheduling 문제

영상별 분산 인코더 효과가 적은 경우



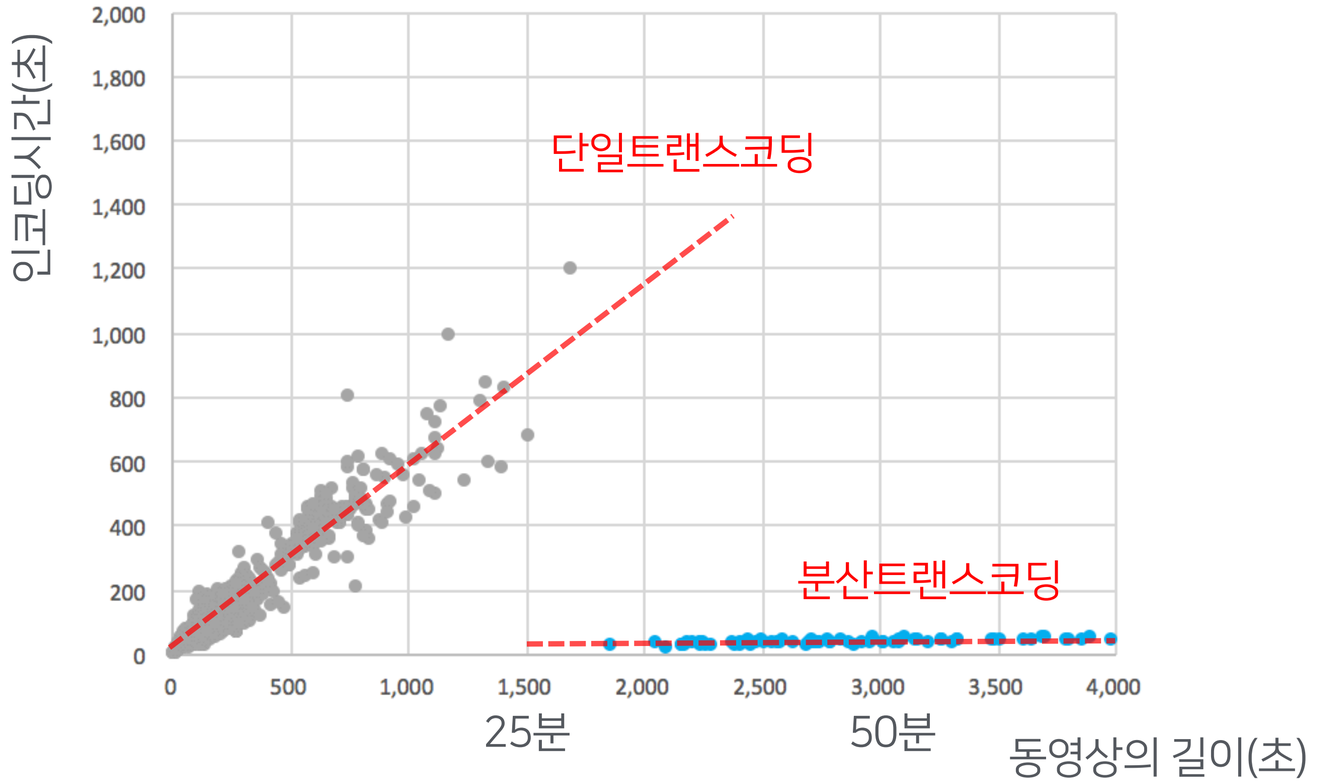
적용 이후 효과는?

배포 시간의 단축 2시간 -> 30분내 완료

5시간 이상 짜리 동영상이 3분내 처리

720P 해상도는 기존 대비 42배 이상 빠름

* 30분이상영상에 적용시



DEVIEW
2019

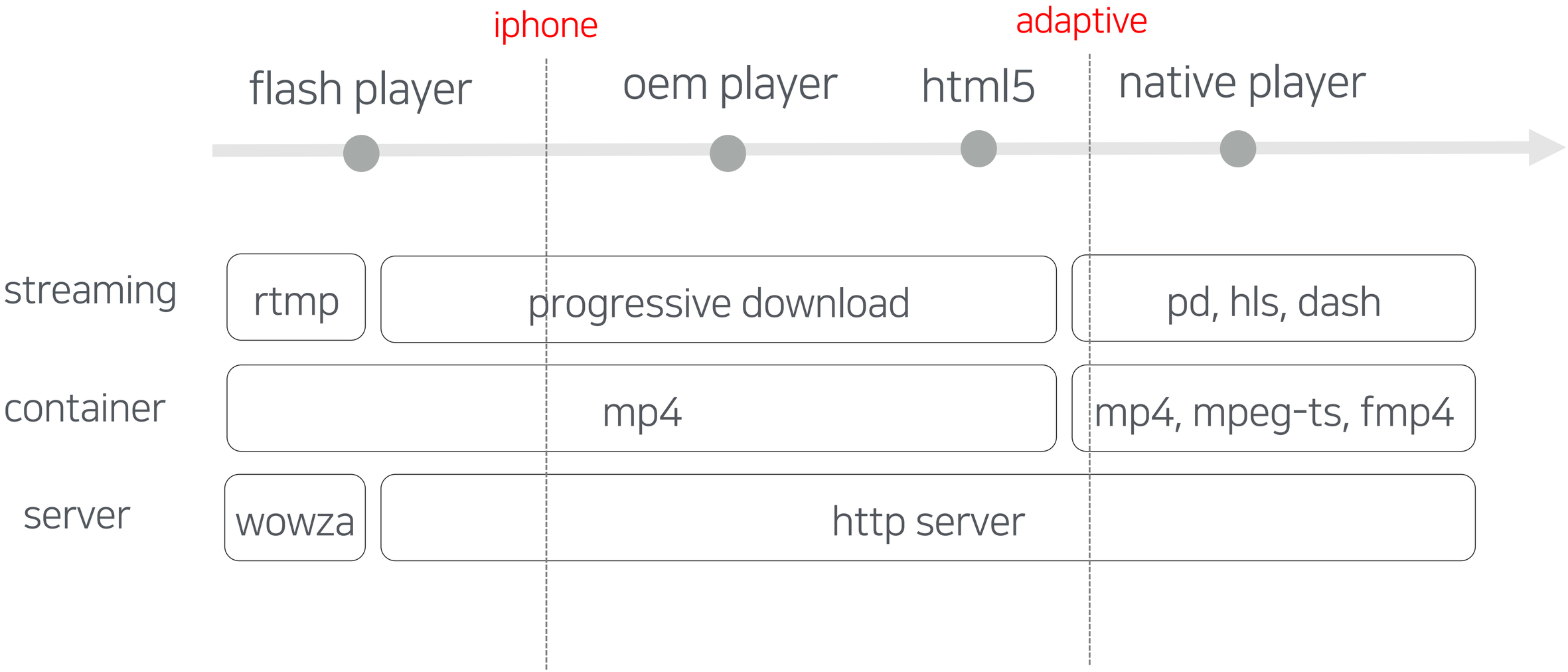
3.안정적인 스트리밍 서비스 개발하기

재생에 필요한 요소와 문제들

재생에 필요한 다양한 정보를 전송할 공통적인 규약이 없다.

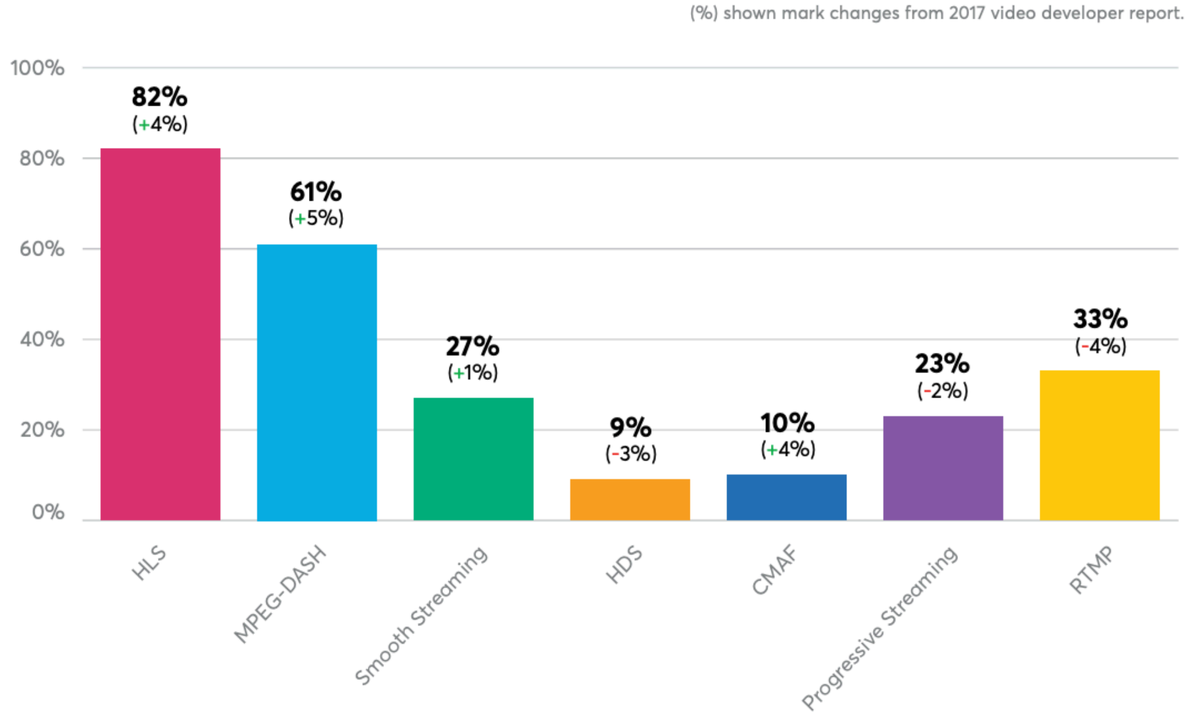
비디오/오디오 스트림, 코덱, 해상도, 자막, 재생 시간, 광고, 360VR, Loudness…

VOD 재생 플랫폼의 변화

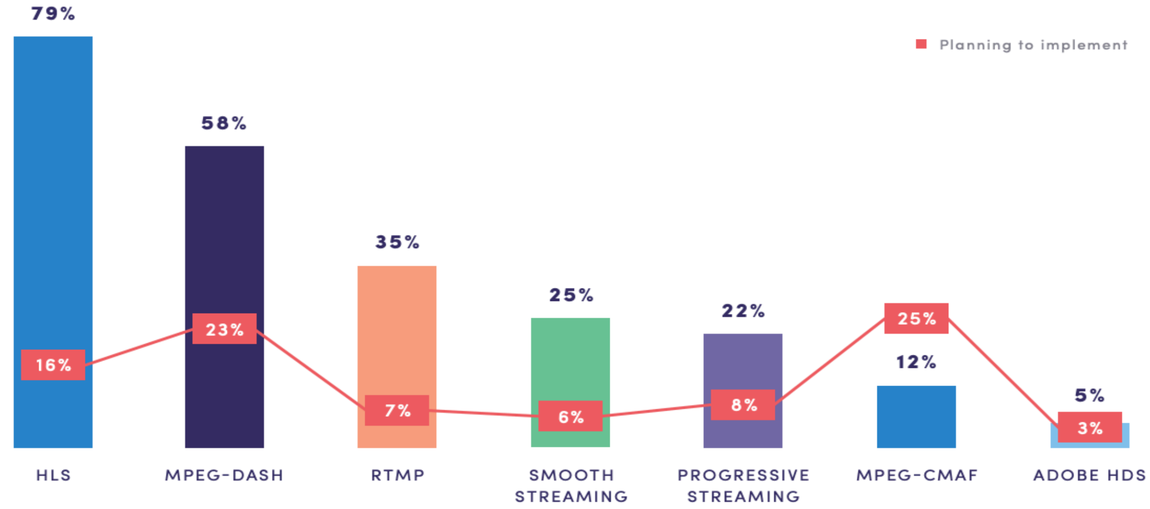


Global Streaming 현황

DEVIEW 2019



2018



2019

[출처] www.bitmovin.com

왜, HLS를 많이 사용하나?

벤더별 Streaming Protocol 파편화

Apple Device의 지속적인 인기

HLS를 지원하는 벤더가 늘어남

그럼, 모든 환경에서 가능한가?

DEVIEW 2019

Android는 3.0부터 Protocol version 2 지원

Network protocols

The following network protocols are supported for audio and video playback:

- RTSP (RTP, SDP)
- HTTP/HTTPS progressive streaming
- HTTP/HTTPS live streaming draft protocol:
  - MPEG-2 TS media files only
  - Protocol version 3 (Android 4.0 and above)
  - Protocol version 2 (Android 3.x)
  - Not supported before Android 3.0

Chrome, Firefox, Edge and IE not (properly) supporting the MPEG-TS

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Br
				3.1-5.1				2.1-2.3							
6-10	12-17	2-68	4-76	6-12.1	10-60	3.2-12.3		3-4.4.4	12-12.1				4-9.2		
11	18	69	77	13	62	13.1	all	76	46	76	68	12.12	10.1	1.2	7.1
	76	70-71	78-80	13.1-TP											

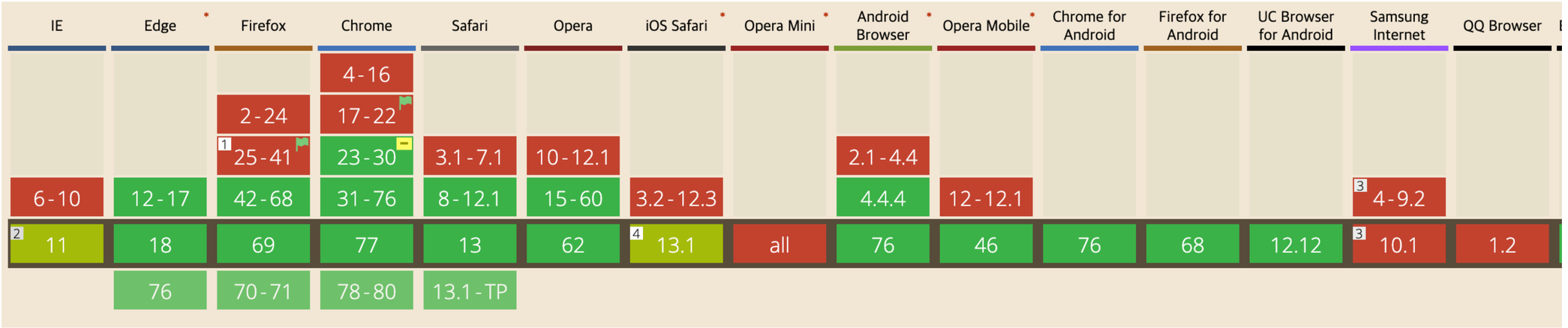
이를 해결하기 위한 방법들

DEVIEW 2019

Android's MediaCodec 구현 (ExoPlayer…)

HTML5 + Media Source Extensions (HLS.js, DASH.js…)

문제는 Native, MSE 도 안되는 환경이 존재



우리는 어떤 Streaming을 사용해야 할까?

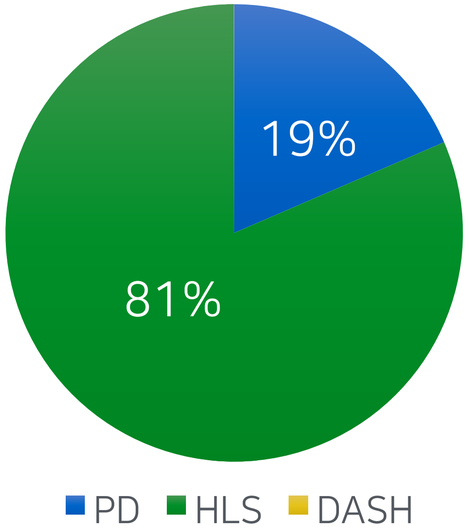
재생 안정성, 넓은 커버리지

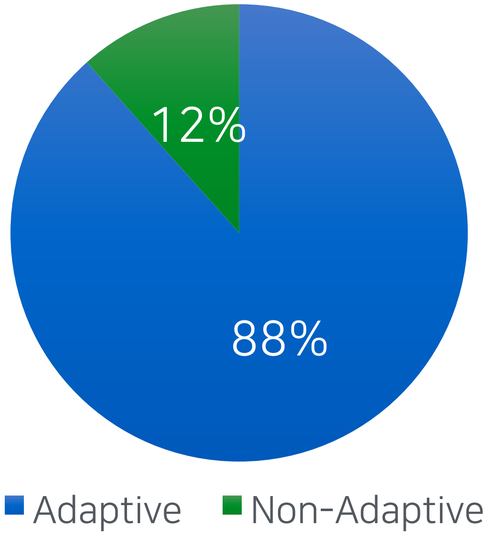
Codec 호환, Adaptive Streaming

수익화를 위한 컨텐츠 보호 (DRM)

낮은 개발 및 운영 비용

PLAYBACK	DRM (CENC)
HLS/PD	MPEG-DASH





Multi Streaming 지원시 고려 사항

플레이어와 재생 시스템의 구조

Protocol 간의 컨테이너 호환성?

Dynamic, Static Packaging?

스토리지 용량, Legacy 영상들의 마이그레이션?

Streaming 방식과 메타 정보

미디어 스트림 + 매니페스트 구조

상호간의 구조적인 호환성은 없음

- 미디어 스트림 Only
- 별도 전달 해야 함

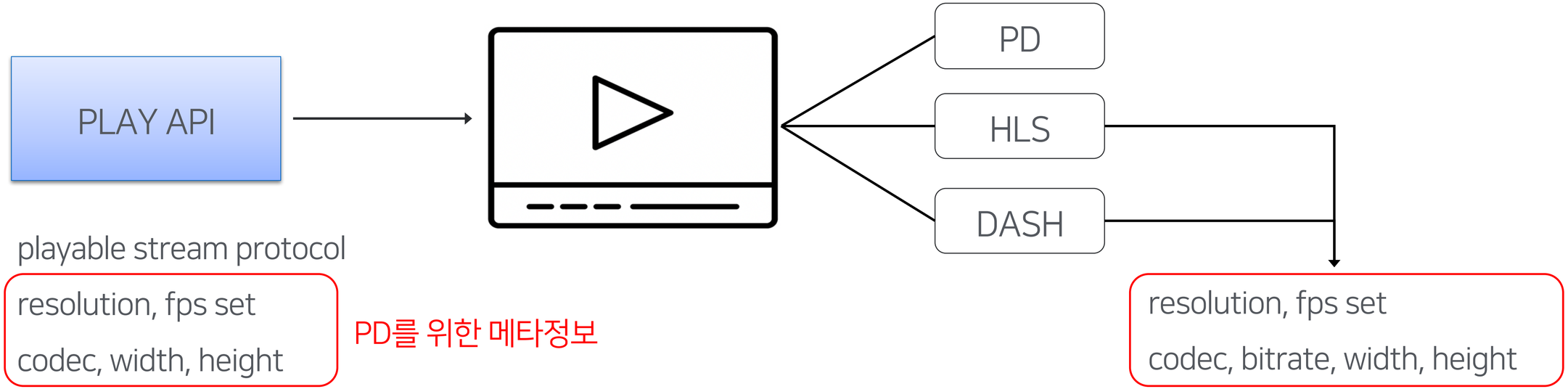
- 미디어 스트림 + 메타
- version에 따라 다름

- 미디어 스트림 + 메타
- 기술 표준, 넓은 표현

Playback 시스템의 구조

API가 PD의 manifest 역할도 제공 해야 함

플레이어는 개별 파서를 모두 구현



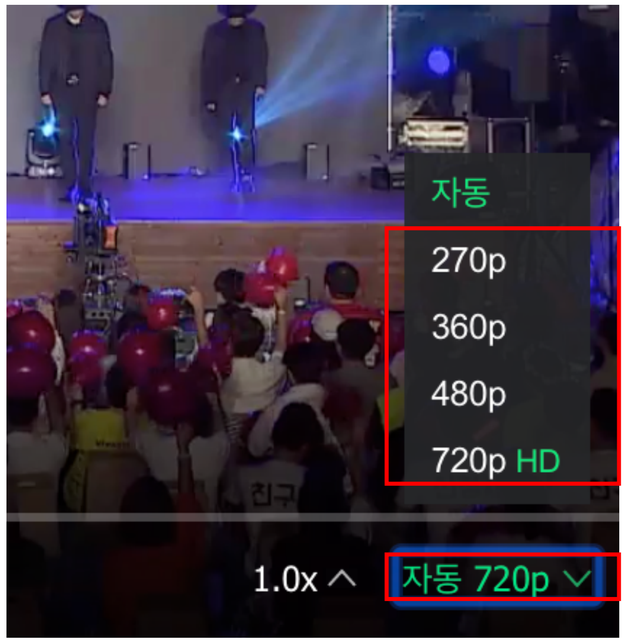
재생중인 화질 표현의 문제 (HLS)

RESOLUTION을 파싱, 세로 영상은?
보다 명시적인 방법이 필요하다.

RESOLUTION

The value is a decimal-resolution describing the optimal pixel resolution at which to display all the v

The RESOLUTION attribute is OPTIONAL but is recommended if the Variant Stream includes video.




```
If (resolution > 720 && resolution <= 1080 ) {
   display_resolution = 1080P
}
```

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-STREAM-INF:BANDWIDTH=4349952,RESOLUTION=1900x1060
83a9ecca-d852-11e9-8b17-246e963a41ed.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=2371584,RESOLUTION=1280x720
89f19736-d852-11e9-bd3f-246e963a49b9.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=1282048,RESOLUTION=854x480
7a3ee1ec-d852-11e9-946c-246e96398ca5.m3u8
```

보다 나은 방법이 있다면 (HLS)

메니페스트의 메타 정보를 확장

GROUP-ID를 지정하고, NAME 으로 매핑

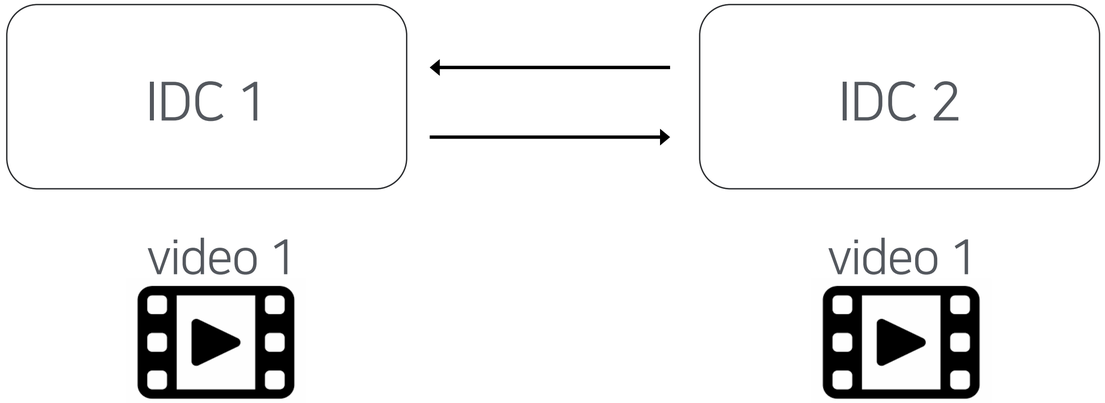
실제 해상도가 달라도 렌더링 해상도 맞추기 용이

```
#EXTM3U
#EXT-X-VERSION:4
#EXT-X-MEDIA:TYPE=VIDEO,GROUP-ID="144_normal",NAME="144"
#EXT-X-MEDIA:TYPE=VIDEO,GROUP-ID="270_normal",NAME="270"
#EXT-X-MEDIA:TYPE=VIDEO,GROUP-ID="360_normal",NAME="360"
#EXT-X-MEDIA:TYPE=VIDEO,GROUP-ID="480_normal",NAME="480"
#EXT-X-MEDIA:TYPE=VIDEO,GROUP-ID="720_normal",NAME="720"
#EXT-X-MEDIA:TYPE=VIDEO,GROUP-ID="1080_normal",NAME="1080",DEFAULT=YES
#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="acc",NAME="audio"
#EXT-X-STREAM-INF:BANDWIDTH=2779136,VIDEO="1080_normal",RESOLUTION=1920x960
fe39fe36-e4ce-11e9-8425-246e96398ca5.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=1630208,VIDEO="720_normal",RESOLUTION=1280x640
f2174dd1-e4ce-11e9-a4d6-246e963a41ed.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=988160,VIDEO="480_normal",RESOLUTION=854x427
ea18f2b7-e4ce-11e9-8425-246e96398ca5.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=835584,VIDEO="360_normal",RESOLUTION=640x320
da414fdf-e4ce-11e9-ac0d-246e963a49b9.m3u8
```

24/7/365 재생 서비스를 보장하자

IDC, Cloud 서비스 장애, CDN 설정 실수 => 적어도 1년에 한번

여러 서비스에 영향이 큰편, 스트리밍 서비스 이중화 전략이 중요

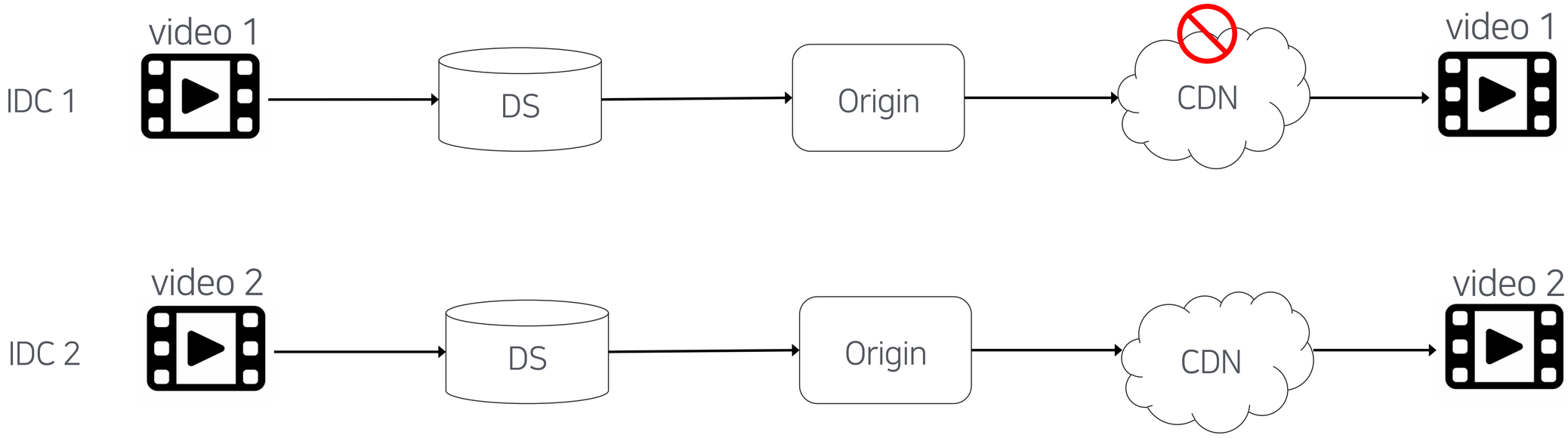


장애시 Fail-over가 가능해야 함

Single-IDC,Single-CDN 에서

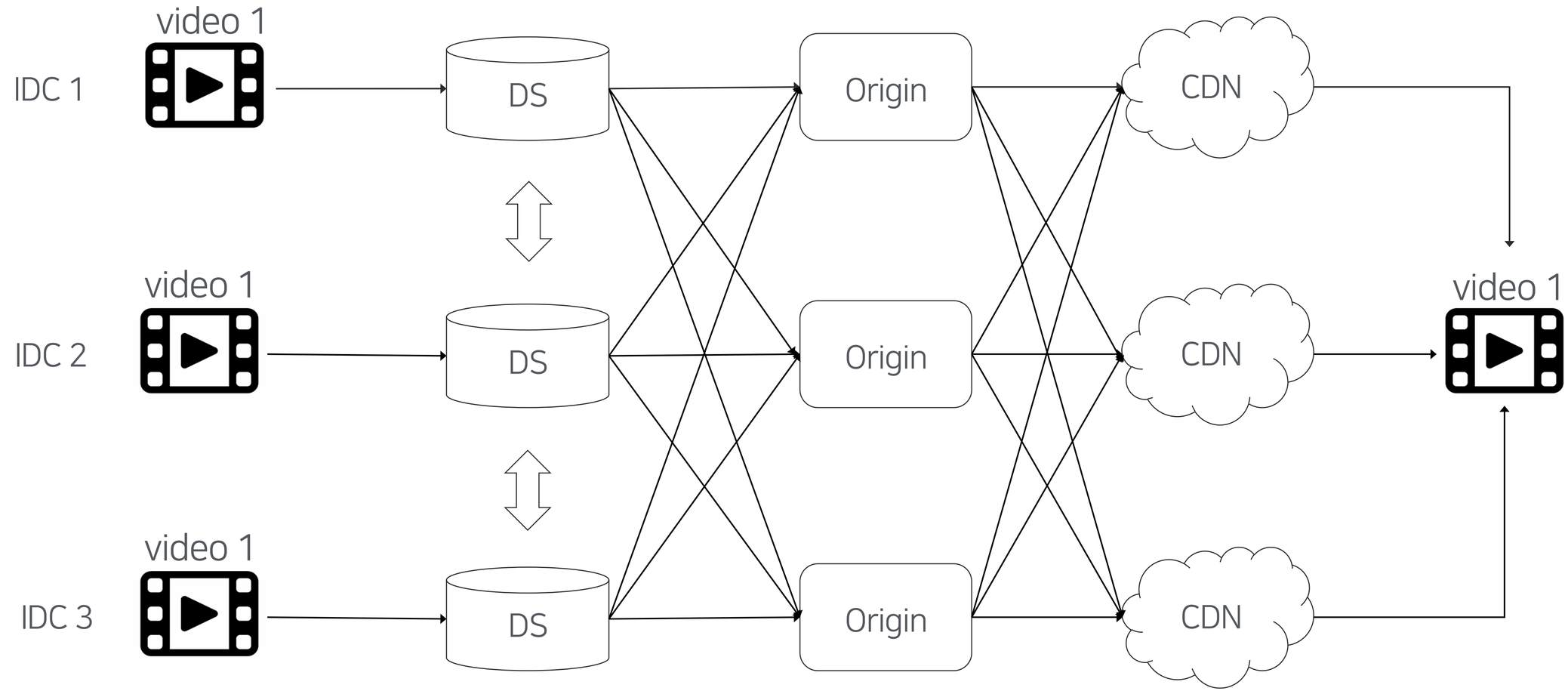
DS(Data Source)-Origin-CDN은 1:1로 매핑된 구조

하나라도 장애가 발생하면 video1은 재생 불가

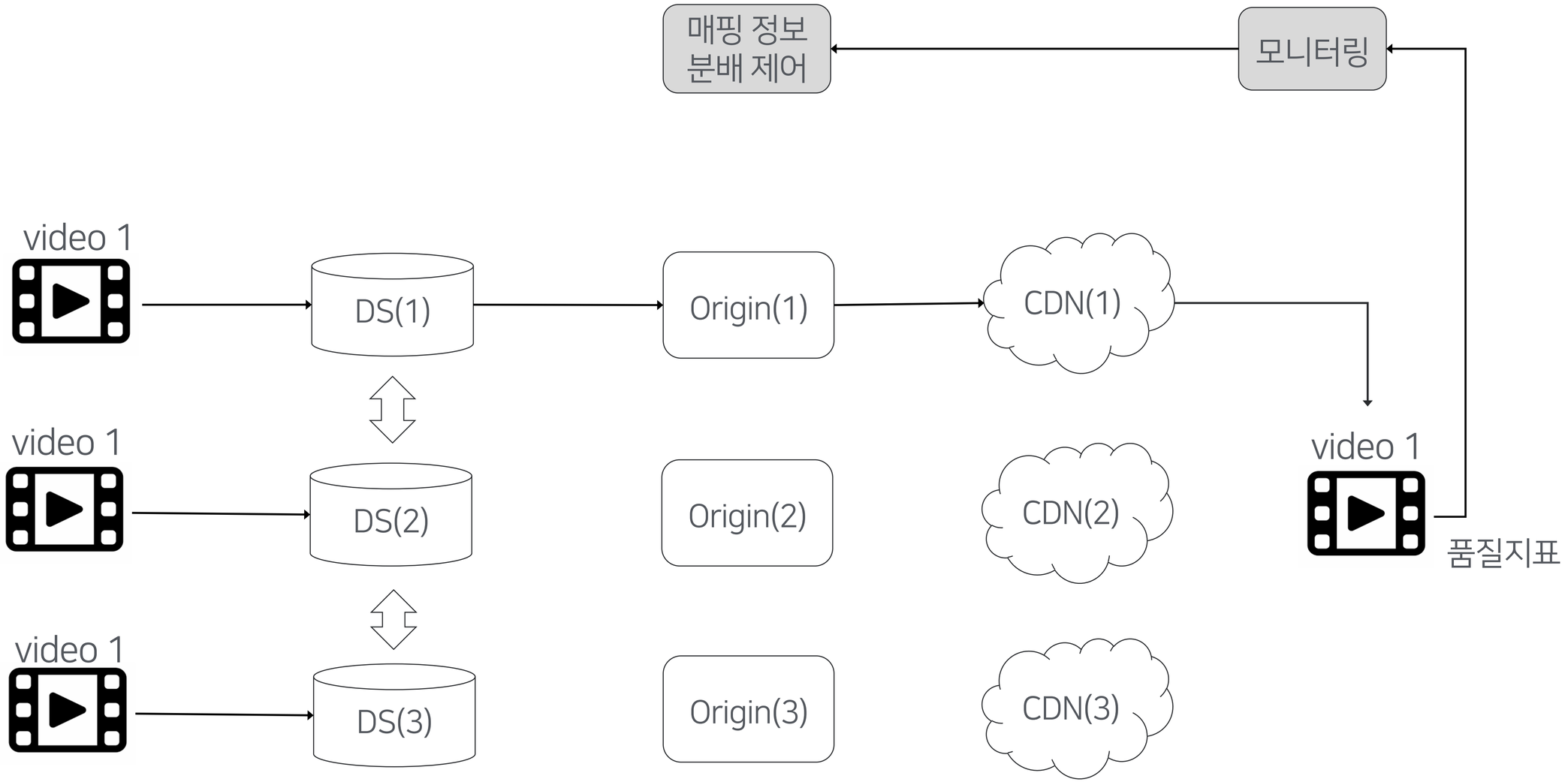


Multi-IDC,Multi-CDN으로

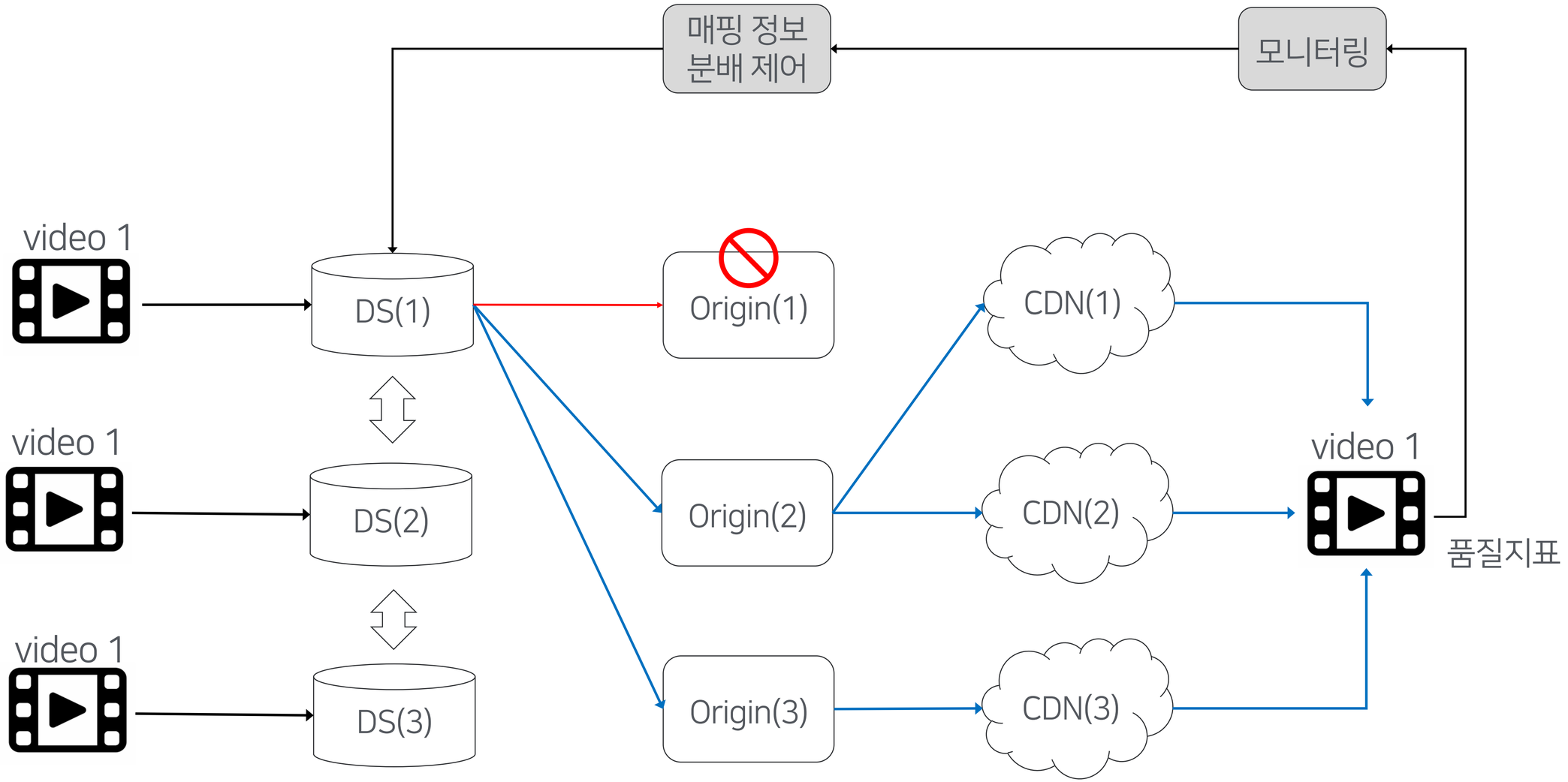
평상시 Locality 를 위해 같은 IDC내 전송 되도록 동작



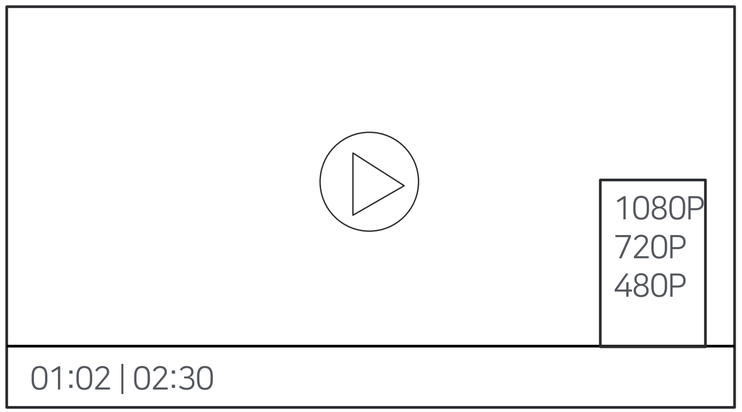
장애 시나리오에서 Fail-Over



장애 시나리오에서 Fail-Over

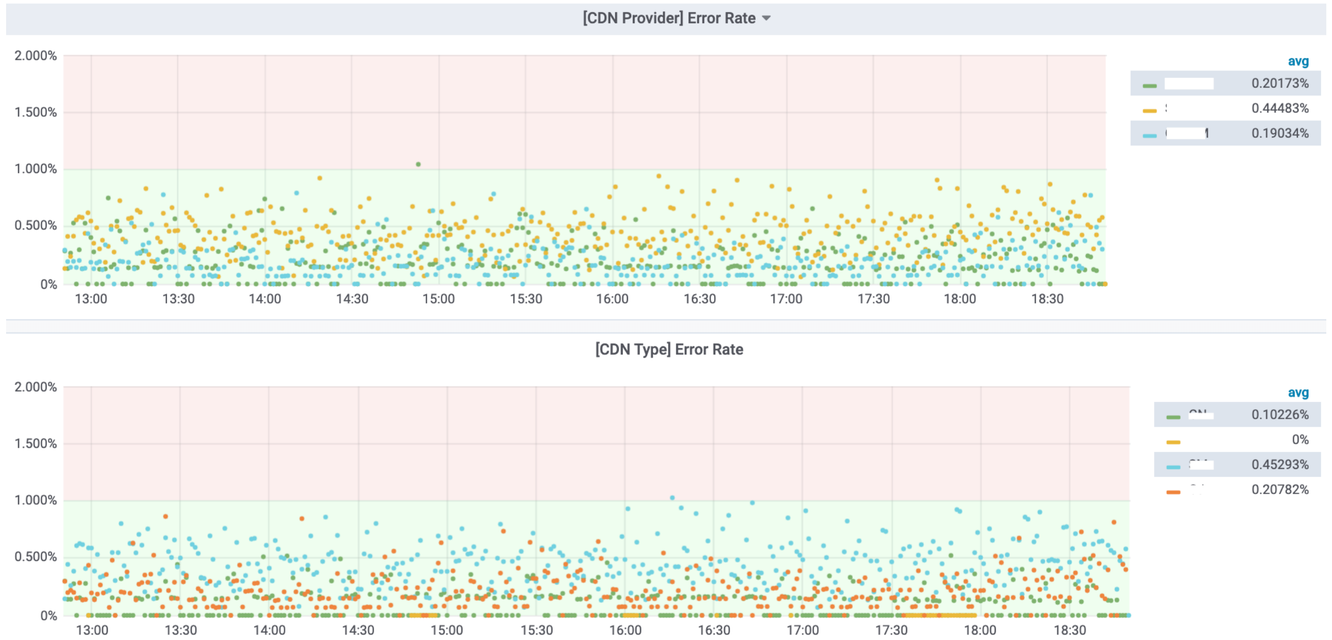


재생 품질 분석 시스템



metrics

- buffering
- Initial time
- api latency
- playback status
- drop frames
- ...



VERT.X → APACHE kafka → APACHE Spark → elasticsearch / hadoop / redis

DEVIEW
2019

4.대용량 플랫폼의 거버넌스

VOD 플랫폼의 운영 현황?

네이버 전사 VOD서비스 제공

수십 PB의 스토리지, 초당 3만건의 트래픽 처리

모듈수 100개, 2,000대 서버, 3개의 IDC에 분산

SpringCloud, Struts, Python, Apache, Nginx, Hadoop, Hive, Druid, Redis, Kafka, Docker, Mysql, MongoDB, AirFlow...

크고 작은 장애에 대한 위험성에 노출

지표 수집과 모니터링

시스템 별 모니터링 지표 정의를 잘해야
데이터 수집 및 시각화
안정, 장애 상황에 인사이트
장애 인지를 위한 알림 시스템

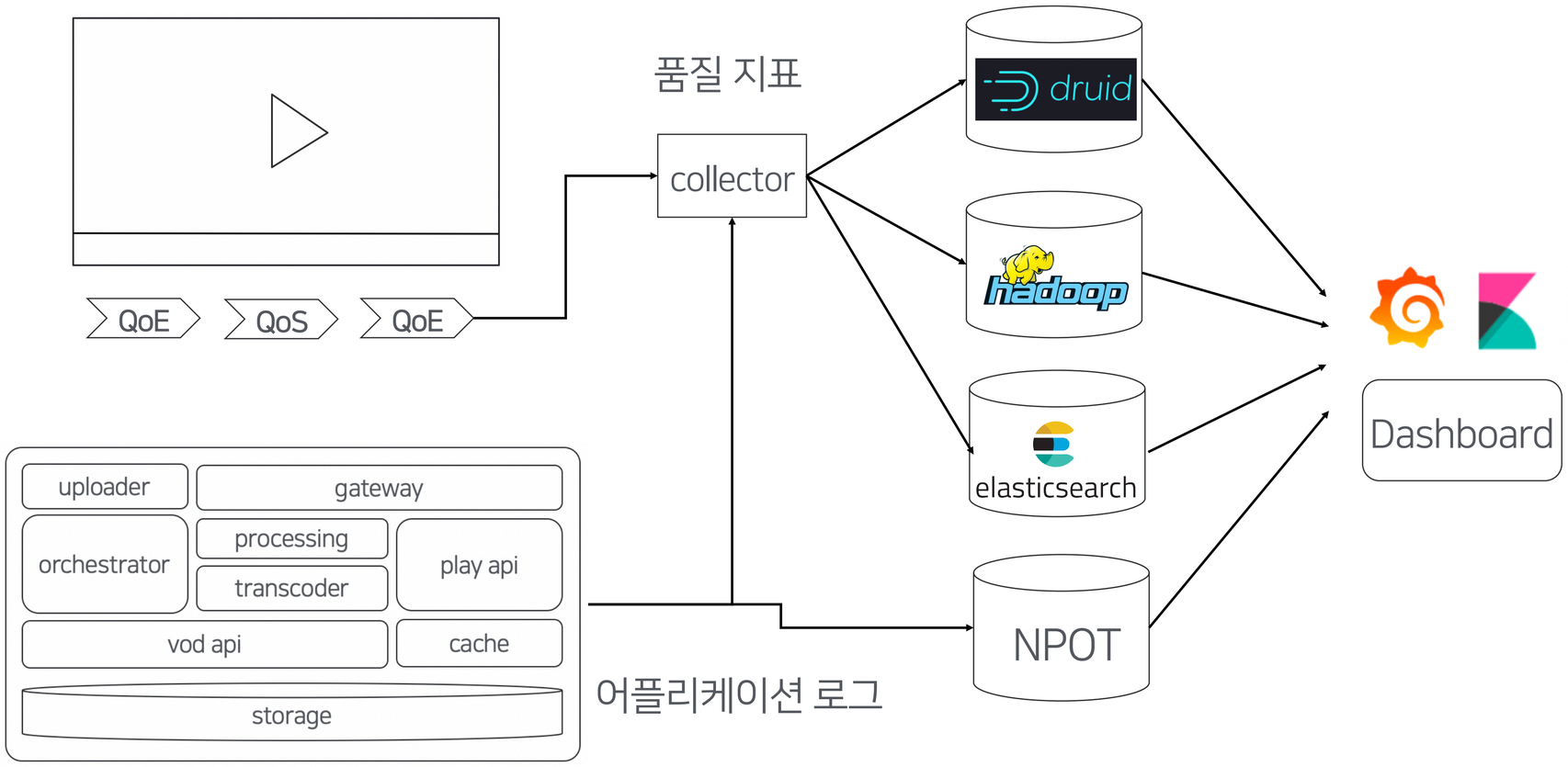
- 현재 사용량은?
- 증설해야할 시점은?

- FS, CPU
- API Latency, Thread

지표 수집과 모니터링

모든 정보를 통합해 보는 시스템?

용도별 특화된 시스템으로 구성하는것이 낫다



재생 품질, 컨텐츠 소비

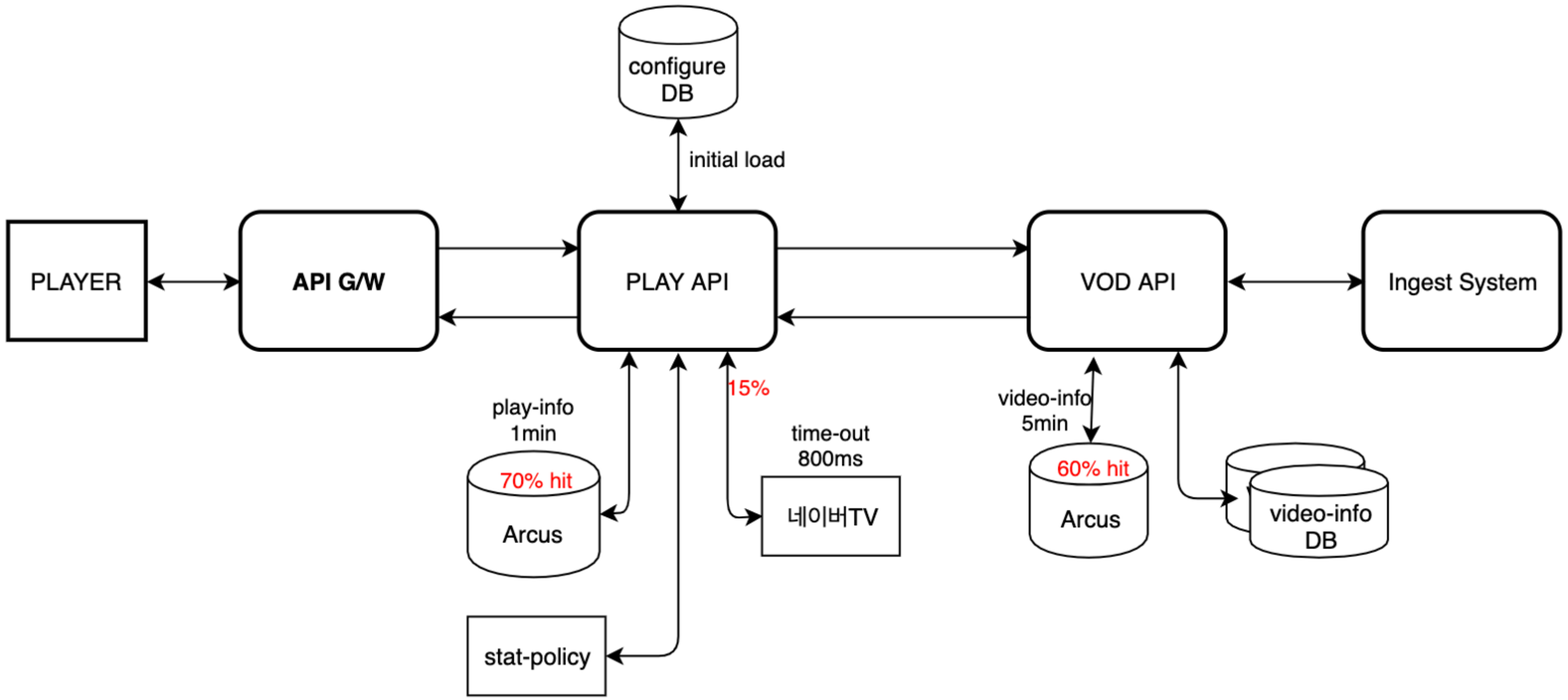
실시간 오류 감지 -> 대응

장애 원인 파악

시스템 가용량 점검 하기

주기적으로 구간별 아키텍처를 갱신 (2회/1년)

미리 구간별 병목 지점 예측



시스템 가용량 점검 하기

DEVIEW 2019

모듈별, 구간별 성능 테스트

가용량 측정표 작성

튜닝, 증설 계획 수립

Server Spec.

Type	Ncloud
Zone	NAVER-CHUNCHEON
Spec	1vCPU, 2GB Mem, 100GB Disk
OS	CentOS 7.3 64Bit

JVM Options

Option	Description
-server	서버 모드로 JVM 실행.
-Xms1g	최소 힙 1g
-Xmx1g	최대 힙 1g
-XX:+UseG1GC	G1 GC 사용

TPS

Result

TPS	MTT	에러율	총 Vuser
2,293.8	21.5	0%	50
2,359.2	42	0%	100
1,977.8	100.5	0%	200
2,262.8	175.5	0%	400
2,131.4	364.8	0%	800

능하면 20ms 이내로 함.

Region의 35%가 찼을 때부터 Mark 함. 기본값은 45

	TPS (current)		예상치		TPS (available)			
서버대수	1대	총합	요구치	최종	1대	총합	예상치대비 가용량	증설계획
50	50	2,500	8,000	10,500	1,200	60,000	5.7	
15	245	3,675	2,400	6,075	7,000	105,000	17.3	
35	80	2,800	11,000	13,800	8,000	280,000	20.3	
N/A		3,300	10,560	13,860	N/A	33,000	2.4	
N/A		13,000	2,400	15,400	N/A	130,000	8.4	
11	310	3,410	2,400	5,810	7,000	77,000	13.3	
40	125	5,000	2,000	7,000	1,000	40,000	5.7	
10	130	1,300	7,200	8,500	1,000	10,000	1.2	7.00
4	1,200	4,800	15,200	20,000	10,000	40,000	2.0	
75	0.4	30	8	38	0.8	60	1.6	19.20
		80	102.4	182.4		220	1.2	144.8

장애 관리 프로세스

장애 리포트

타임 라인 (복구 시간)

원인 찾기

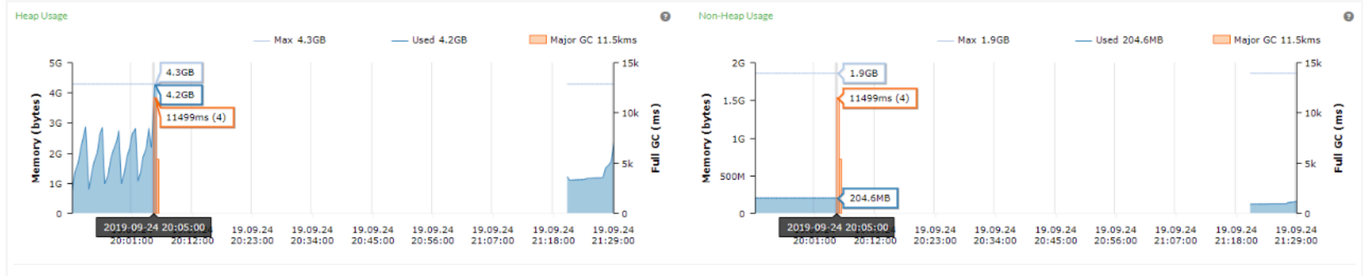
결과 리뷰

타임라인

- 20:05 X001,X003 서버 응답 지연. 장애 발생
- 20:08 X001,X003 port Down 알람.
- 20:40 서버 유레카 제외하려고 하였으나 제대로 동작하지 않아 서버 내림. 오류 줄어듦
- 20:48 실패뜬 영상 복구 시작
- 21:25 특이사항을 찾을 수 없어 X001 서버 재투입
- 21:40 X003 서버 역시 재투입
- 22:24 특별히 원인을 찾을 수 없어서 일단 종료.
- 다음날 원인 분석
  - 힙덤프 분석 : search errors 이후에 에러가 나서 pipeline이 50%정도까지 차지
  - 로그 분석 및 nbaseArc담당자분께 문의

원인

- station ingest monitoring 요청시 url파라미터로 done=true 값을 붙여서 보냄.
- 예전에 한번 호출한 url을 자동완성하여 보낸 것으로 보임.
- orbit에서 ingest Status 정보를 한꺼번에 너무 많이 가져와서 처리하다가 fullGC가 발생하면서 장애 발생



그외 또 다른것들이 있다면…

DEVIEW
2019

플랫폼의 개발 원칙이 있어야 하고

플랫폼을 개발/운영하는것은 크고 작은 결정의 연속

고가용성, 재사용성을 위한 설계, 클라이언트 설정 방지 원칙

자동화에 대한 지속적인 노력이 필요

시스템,서비스가 늘어날수록 자체가 Overhead로 작용

플랫폼 개발 연동 가이드, CI/CD, 모니터링 운영 자동화

What's Next?

딥러닝 기반 트랜스코딩

늘어나는 트래픽으로 인해 인코딩 효율화가 중요

딥러닝 기반, 장면 복잡도에 따른 인코딩 최적화 기술

Immersive 동영상 경험

8K/60fps 트랜스 코딩 기술, VR 서비스 커버리지

DOLBY VISION/ATMOS/HDR+

동영상 메타 DB 서비스

동영상의 다양한 메타 정보를 수집, 분석할 수 있는 플랫폼

등장 인물, 하일라이트 구간, 주요 장면 서비스 제공

DEVIEW
2019

Q & A

DEVIEW
2019

Thank You