5만개 볼륨에서 발생하는 초당 20GB 데이터 저장 기술
(Ceph를 이용한 대규모 스토리지 구축 및 운영)

네이버 스토리지플랫폼 유장선

NAVER

CONTENTS

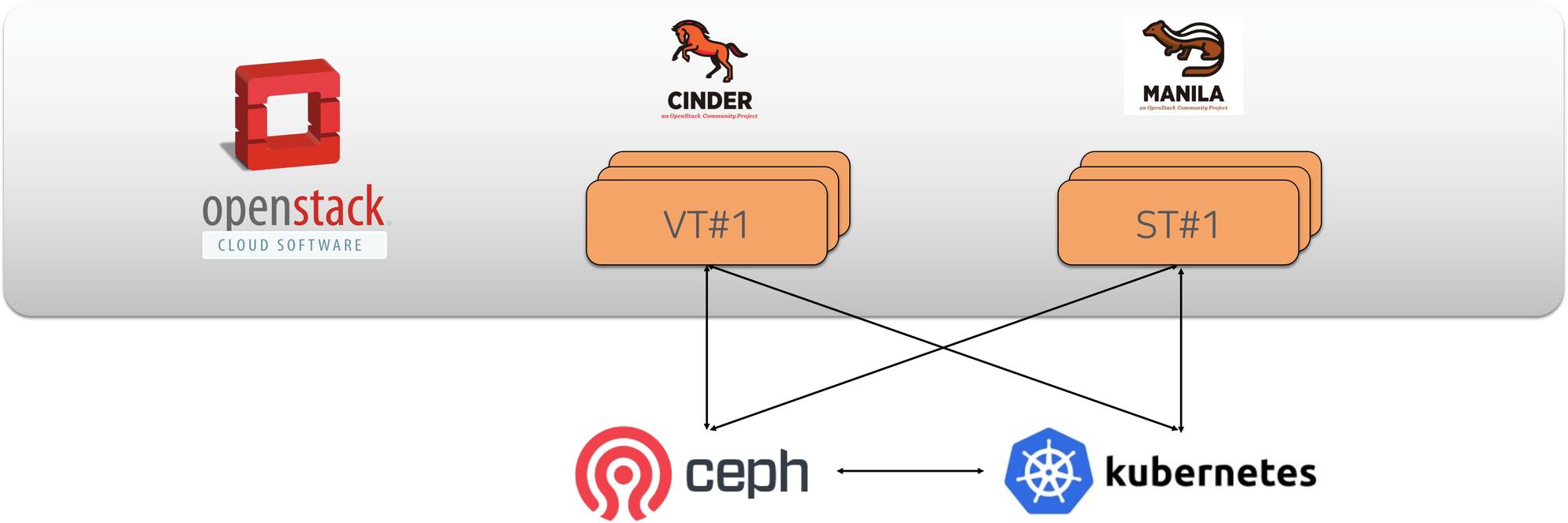
1. 스토리지 구축 및 사용

1.1 클러스터 현황

1.2 OpenStack

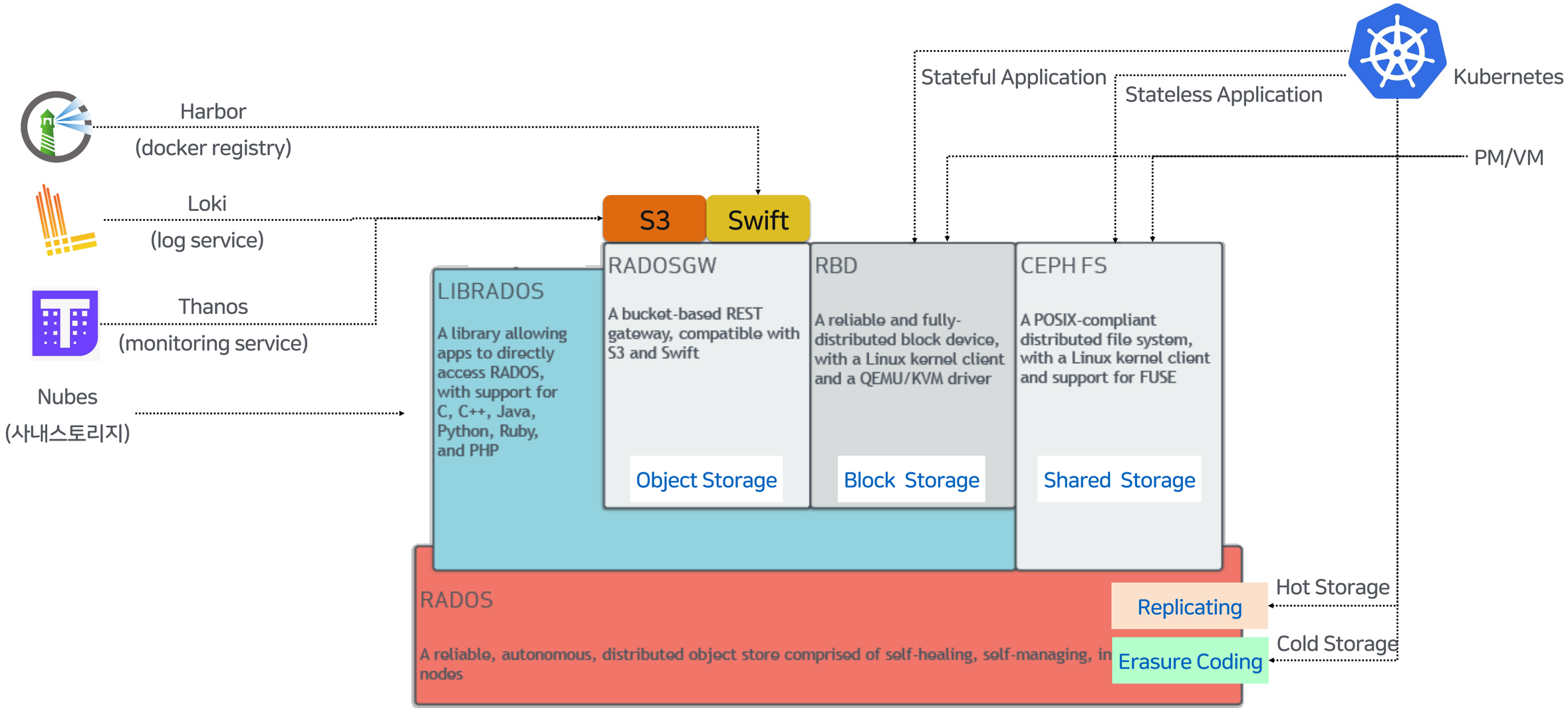
OpenStack과 LDAP을 연동하여 인증 및 Muiti-Tenancy를 제공함

Cinder와 Manila를 활용하여 대규모 클러스터 환경을 제공함



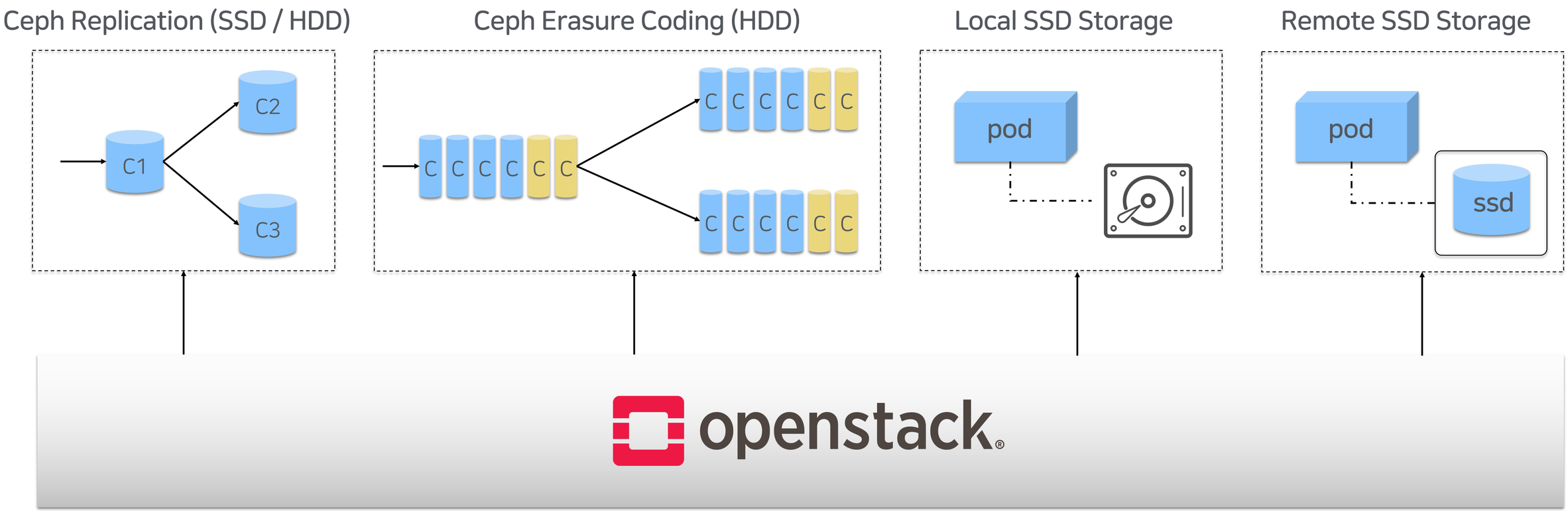
1.3 Ceph Storage

Ceph는 오픈소스 스토리지이며, 하나의 스토리지를 통해 Object, Block, Shared Storage를 모두 제공합니다.



1.4 Multiple Storage Type

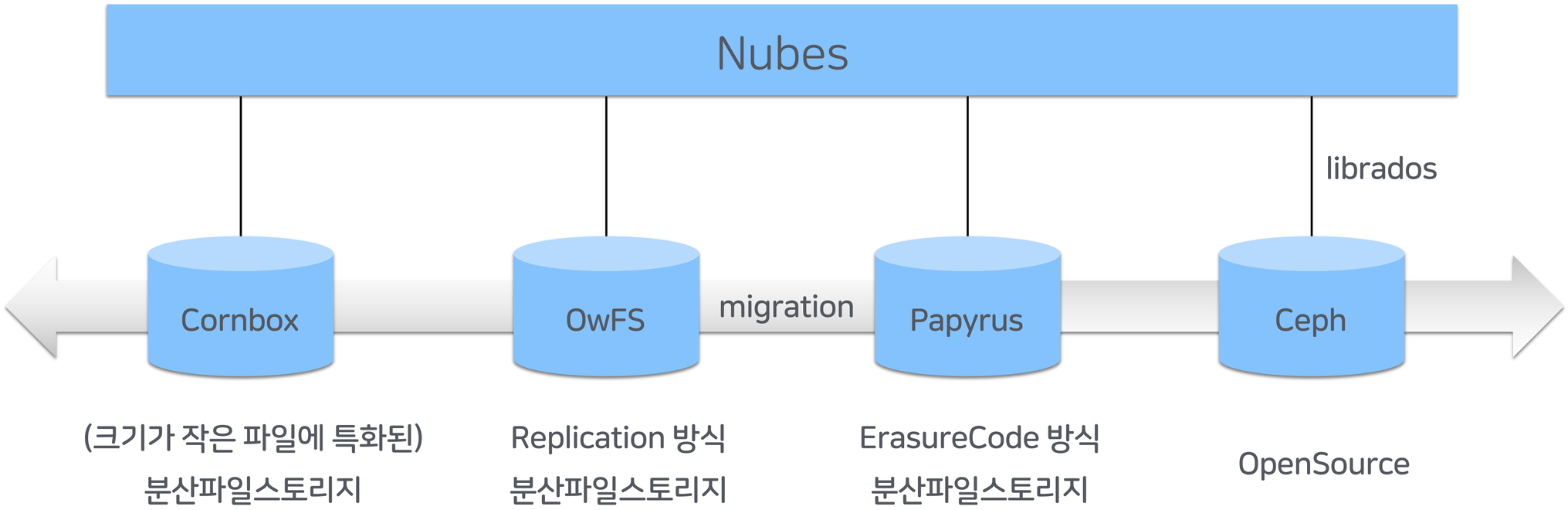
다양한 워크로드를 지원하기 위하여 NVMe/SSD/HDD디바이스로 스토리지를 구성하고,
Replicating / Erasure Coding 구성 및 Local / Remote SSD등을 제공하고 있습니다.



1.5 사내 스토리지 연동

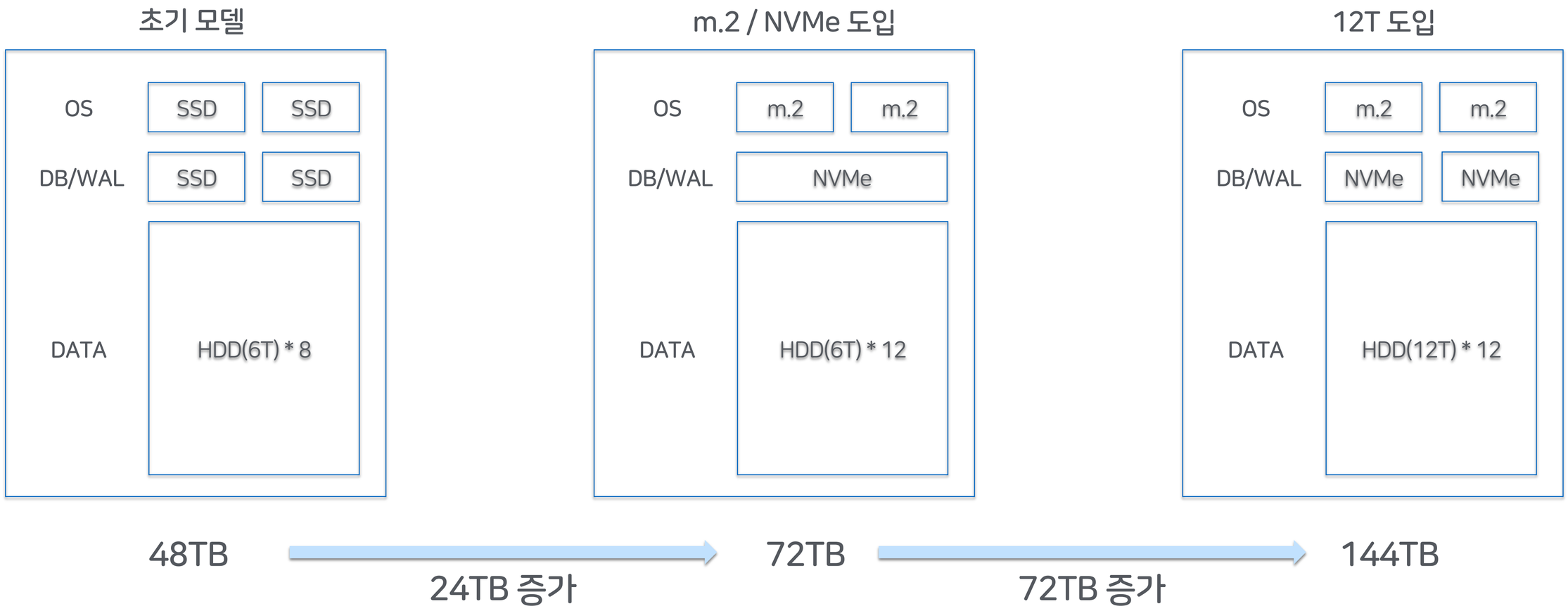
자체 개발된 사내 Object Storage (Nubes) 의 Backend Storage로 제공함

Nubes 를 통해 제한없는 저장 공간 제공 및 서비스 상태에 맞는 스토리지 타입을 제공함



1.6 Node Architecture

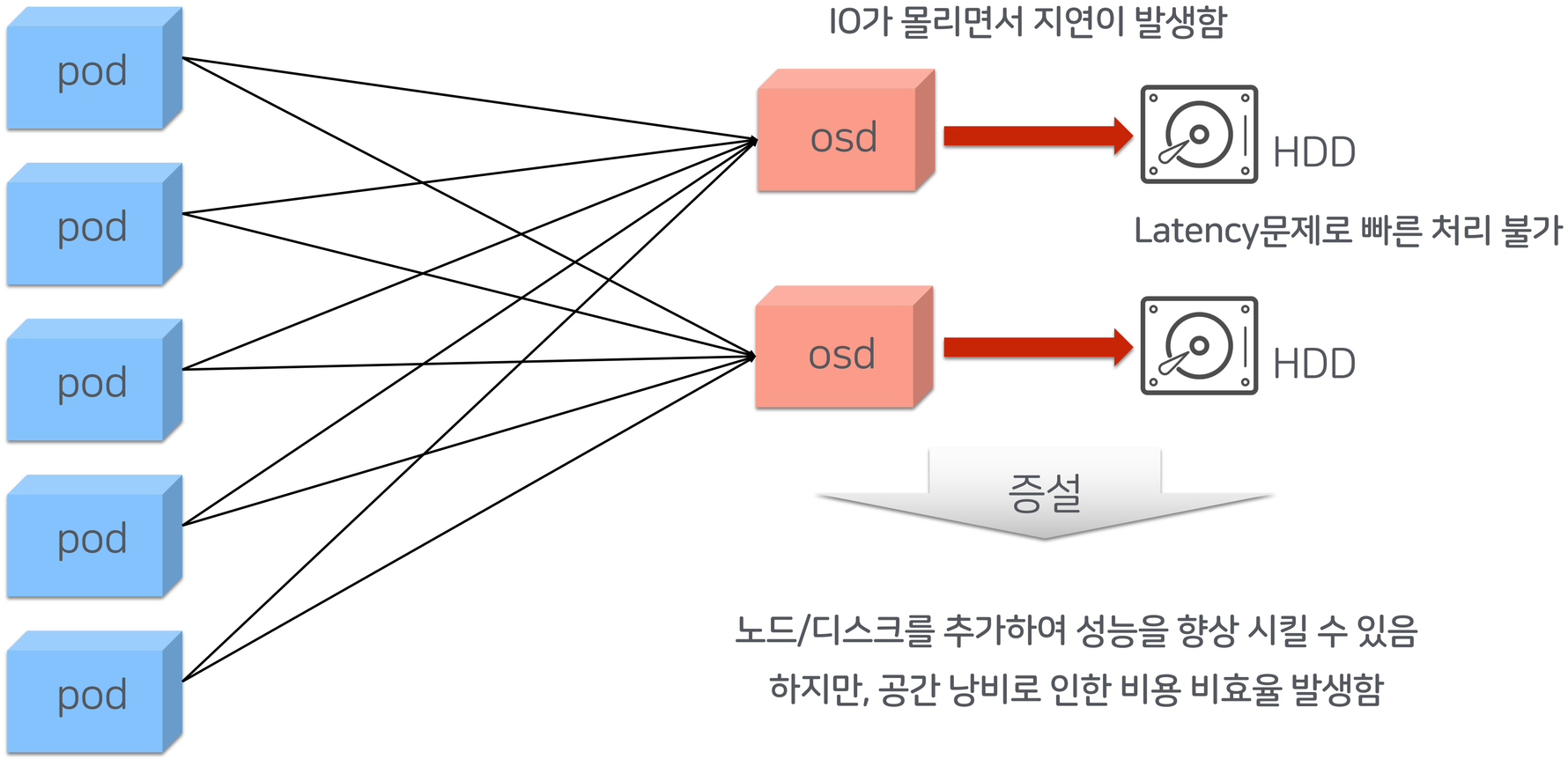
Node Architecture개선을 통하여 GB 당 비용을 줄이고, 성능을 개선함 (12Bay Node)



2. 주요 문제점 해결 사례

2.1 (HDD) 클러스터 성능 이슈

수 십개 Pod로 구성된 서비스에서 새벽에 백업이 동작하여 IO 가 몰리는 경우,
HDD Latency이슈로 인해 서비스 IO 영향이 발생하게됨



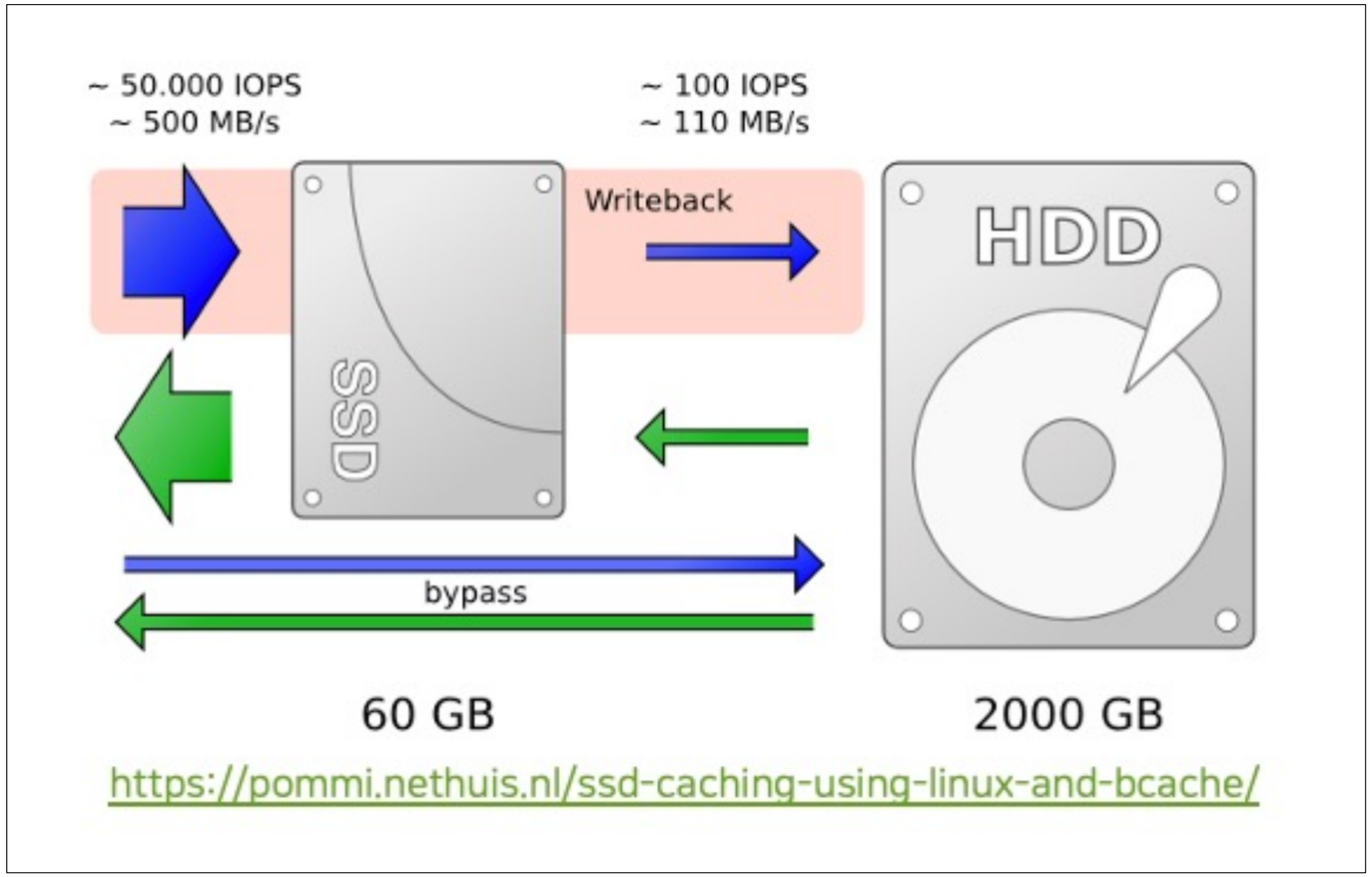
노드/디스크를 추가하여 성능을 향상 시킬 수 있음
하지만, 공간 낭비로 인한 비용 비효율 발생함

2.1 (HDD) 클러스터 성능 이슈

Block Cache는 느린 HDD 디스크의 성능 개선을 위해 앞 단에 빠른 디스크를 두고,
IO를 먼저 처리한 후 HDD로 천천히 write하는 기술입니다.

Random write는 빠른 디스크 처리

Sequential write는 Random Cache를
제거할 수 있기 때문에 bypass 시킴



2.1 (HDD) 클러스터 성능 이슈

Bcache는 kernel 3.10 에 포함되어 있어서, 생성 툴(bcache-tools) 빌드 후 사용 가능함

구성은 쉬운 편이지만, 디스크가 많다면 복잡도는 증가됨

- Kernel Module Enable

$ modprobe bcache

- Install bcache-tools

$ git clone https://evilpiepirate.org/git/bcache-tools.git

$ cd bcache-tools

$ make

$ make install

- Create bcache

$ make-bcache -C /dev/nvme0n1p18 -B /dev/sdh2 –writeback

$ lsblk -o NAME,MAJ:MIN,RM,SIZE,TYPE,FSTYPE,MOUNTPOINT,UUID,PARTUUID | grep bcache

__nvme0n1p18 259:18 0 99G part bcache 4190a4e9-5825-4e2e-8a91-026447b352da 07c25f99-6d71-494b-a044-618d10efa959

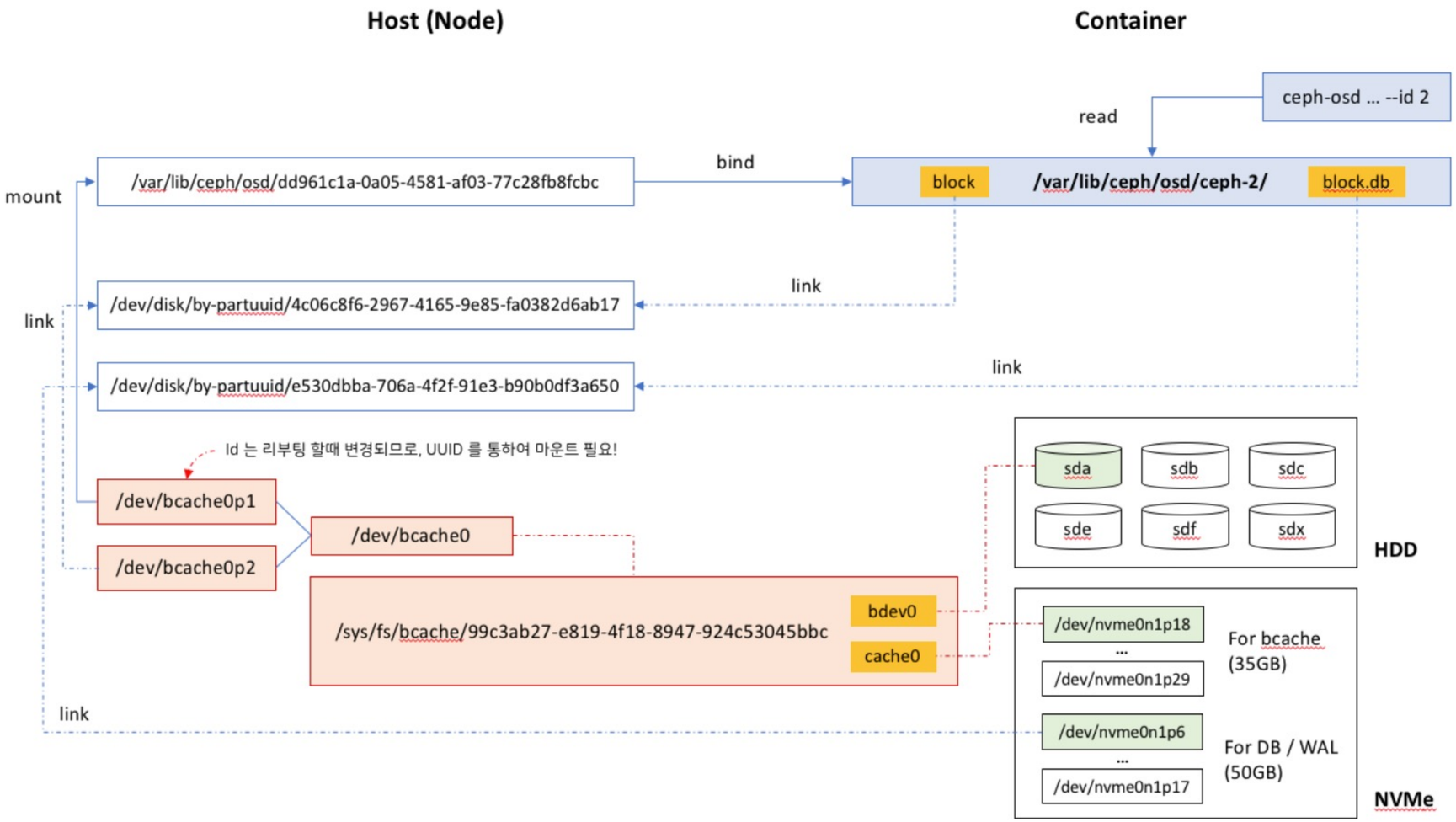
_ __bcache0 252:0 0 5.5T disk

__sdh2 8:114 0 5.5T part bcache fe715aac-f186-43e9-98e7-11bb0eebefa3 e65062e6-05a4-4d10-b718-15928a008f31

__bcache0 252:0 0 5.5T disk

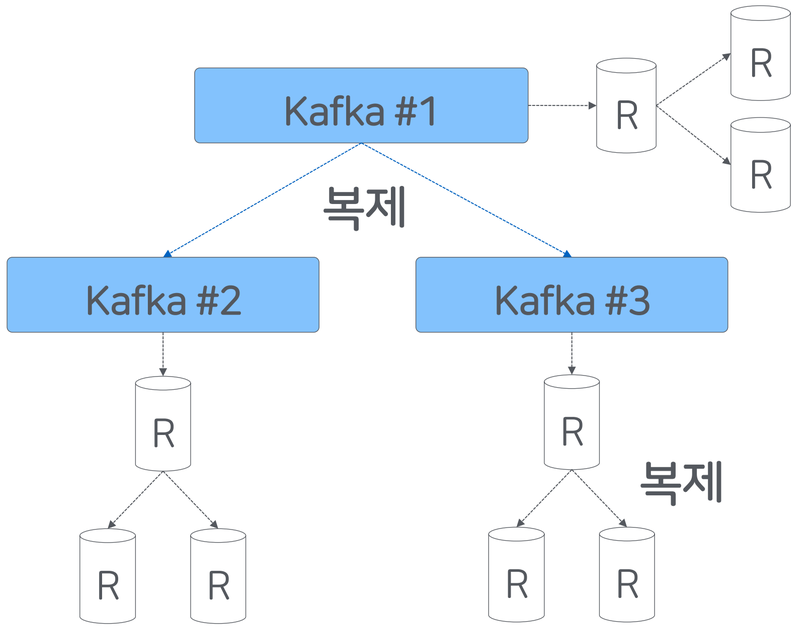
2.1 (HDD) 클러스터 성능 이슈

1개 디스크를 bcache로 구성 시 모습



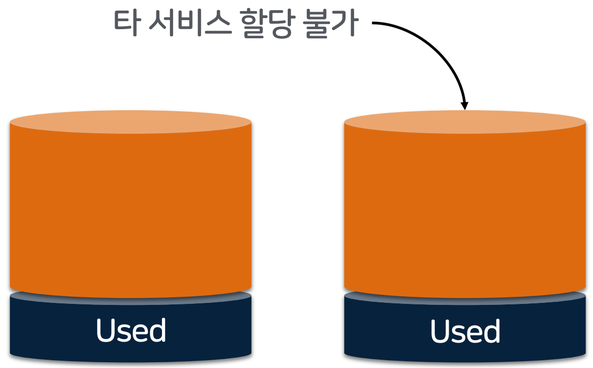
2.2 분산 서비스 on 분산 스토리지 이슈

분산 서비스는 자체 복제방식으로 데이터를 저장하고 있고,
분산 서비스에서 분산 스토리지 사용 시 많은 복제가 발생하게 됩니다.
로컬 디스크는 제한적인 수량과 미 사용시 비용 비효율이 발생합니다.



3 copy * 3 copy -> 9 copy 이슈

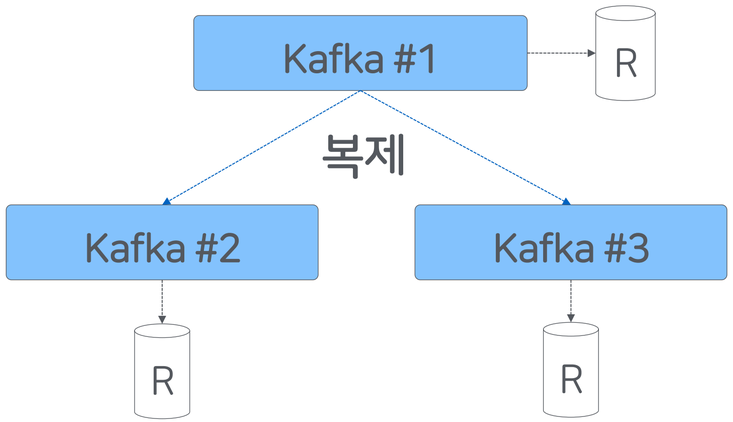
제한적인 로컬 디스크 수



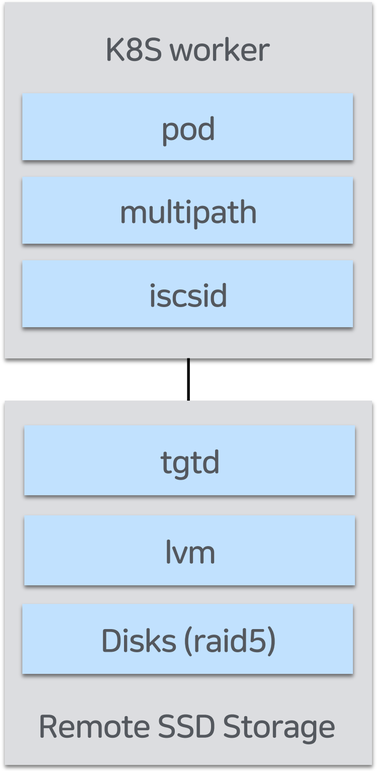
미사용 시 비용 비효율

2.2 분산 서비스 on 분산 스토리지 이슈

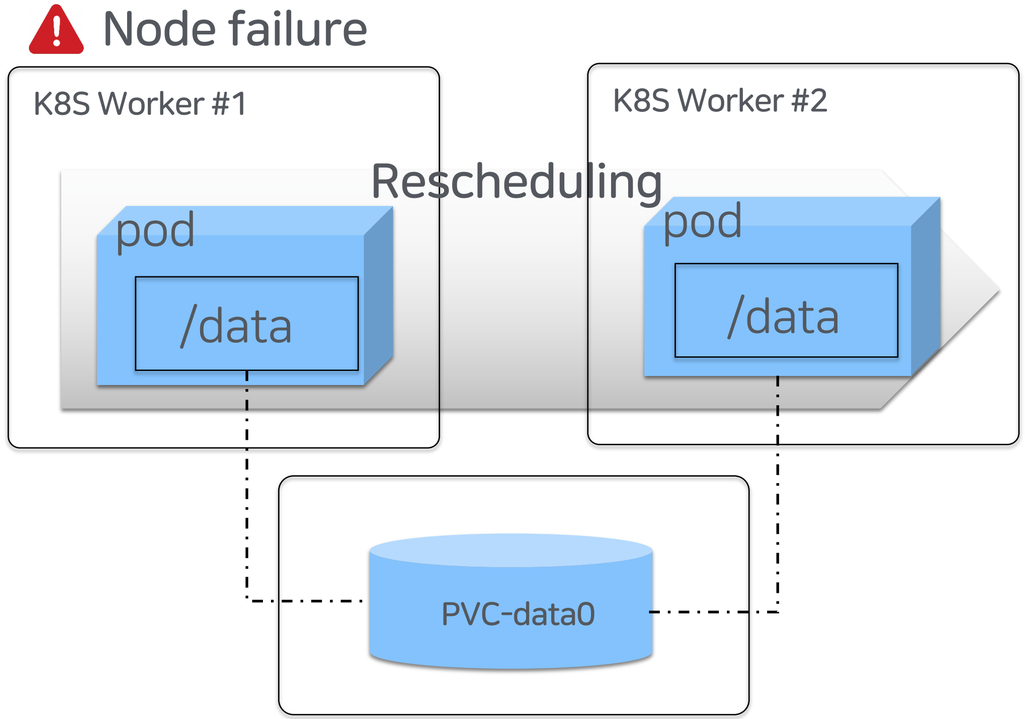
원본만 저장하는 (1 copy) Remote SSD Storage를 제공하여 불필요한 공간 낭비 제거,
Worker Node장애 시 데이터 보존되며, 바로 서비스 재개 가능



1 copy : 추가 복제 없음



필요한 만큼 할당 (낭비 최소화)

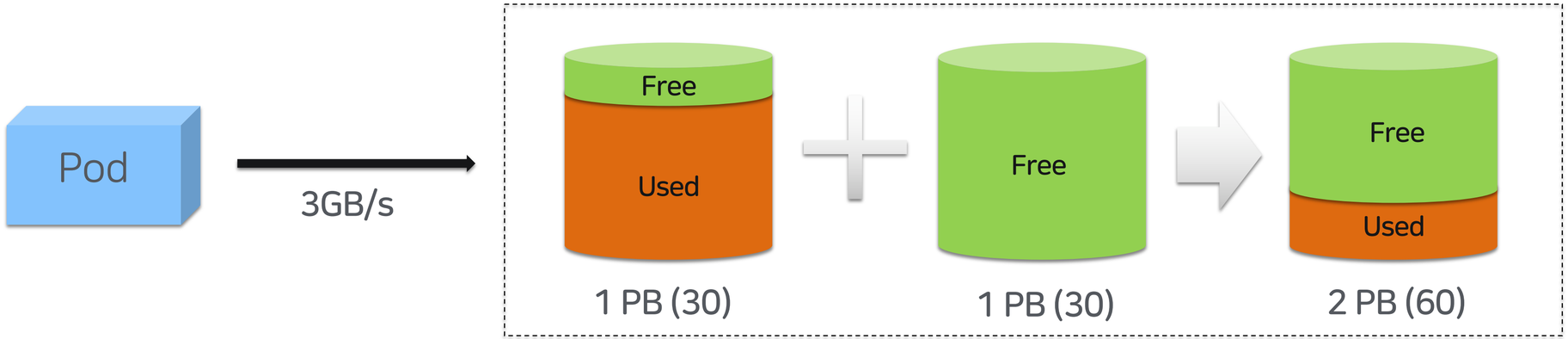


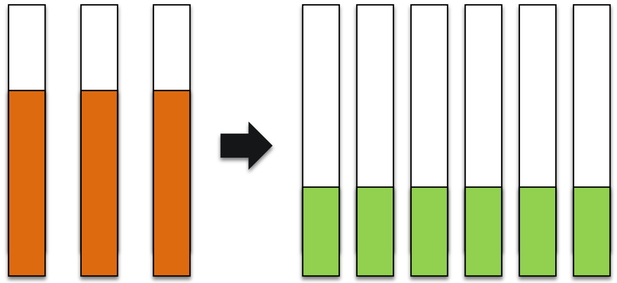
노드 장애 시 데이터는 보존되며,
Rescheduling시 서비스 재개

2.3 스토리지 증설의 현실적인 문제점

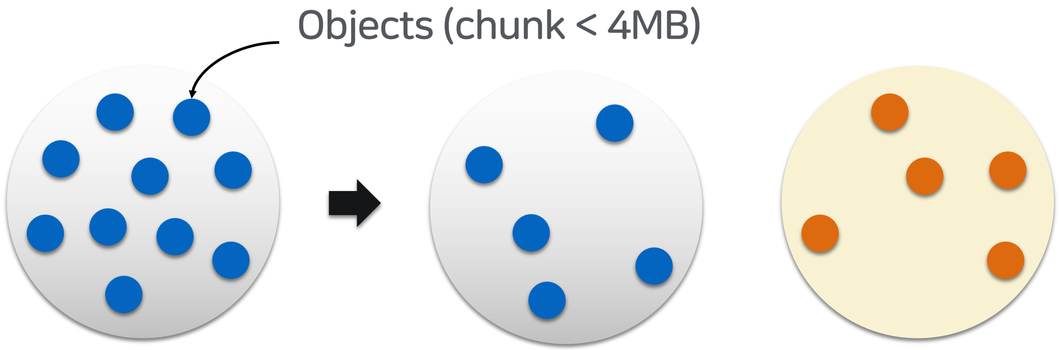
서비스 중에도 증설이 가능합니다.

증설은 노드 증설과 PG (Placement Group) 증가 작업으로 구분됩니다.





노드 증설 (Reallocation)



PG Split (2048 -> 4096)

2.3 스토리지 증설의 현실적인 문제점

증설은 서비스 영향도 증가, 오랜 증설에 따른 운영 리소스 증가, 잦은 재배치로인한 디스크 마모도 증가 문제가 발생함

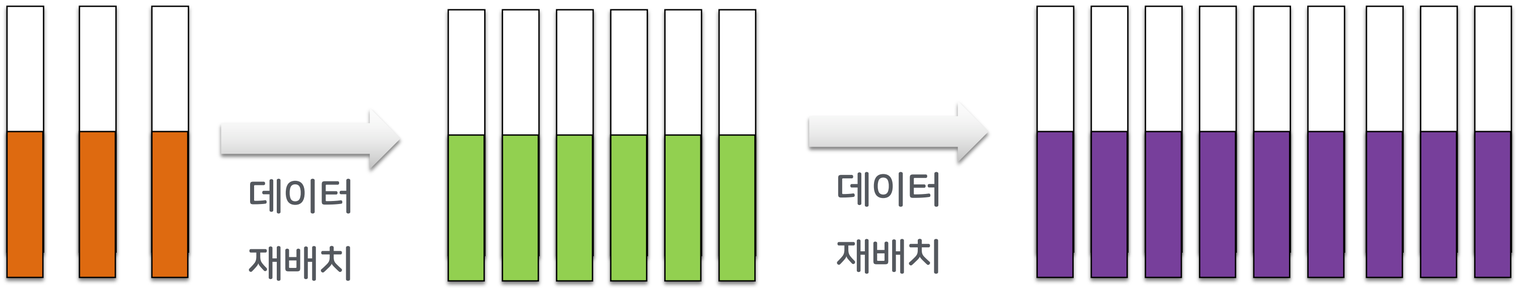
긴 증설 시간
(수개월의 운영 작업)

- 최저 속도로 진행 (osd_max_backfills 1)
- 사용량 높은 낮 시간 중지 (밤에만 진행)

→

- 수 개월 이상 증설 작업 필요
- 쌓이는 데이터 > 확보되는 공간

디스크 마모도 증가

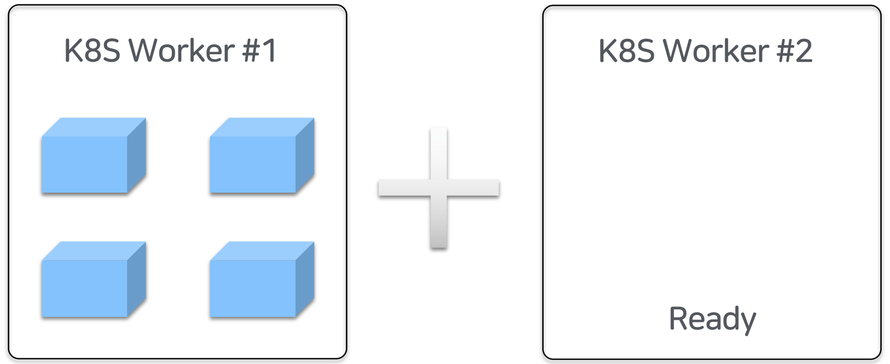


잦은 데이터 재배치 작업으로 read/write 증가에 따른 디스크 마모도 증가

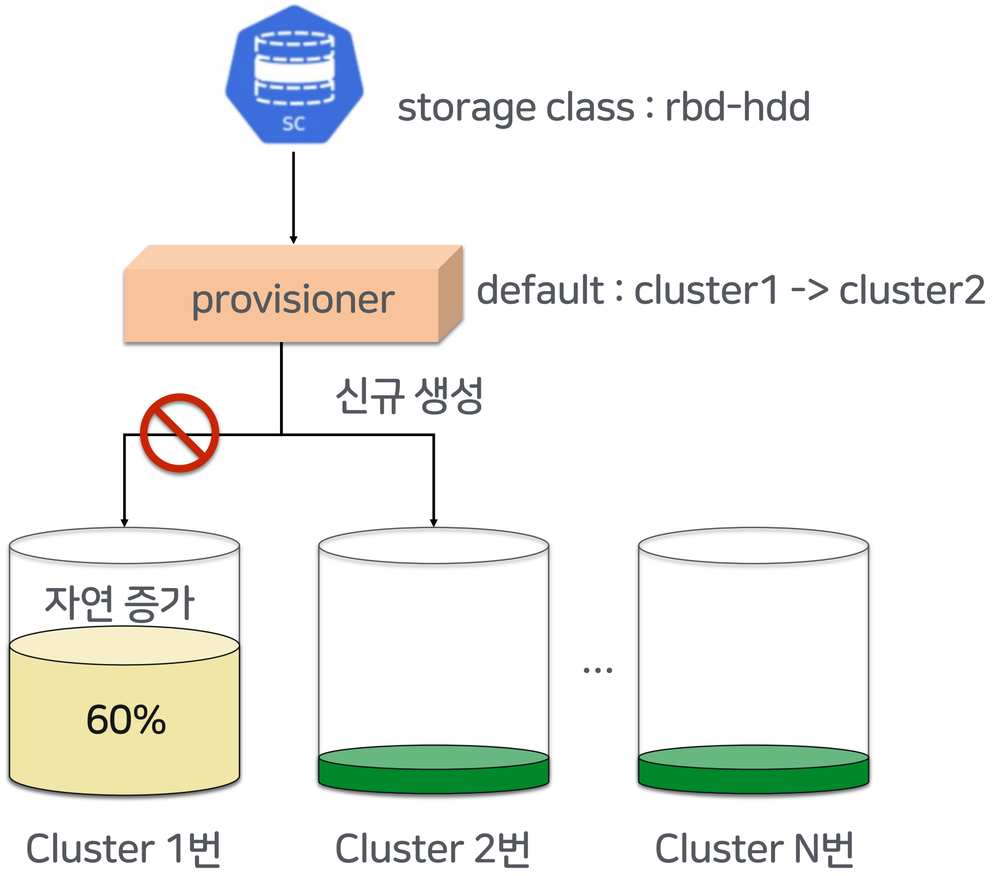
2.3 스토리지 증설의 현실적인 문제점

Compute Node처럼 투입 시 바로 사용 가능한 구조를 Storage 에도 적용함

Provisioner를 통해 볼륨생성 클러스터를 지정하는 방식으로 재배치 없이 증설이 가능한 구조로 변경됨



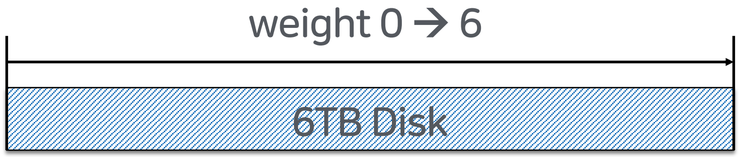
노드 부족 시 신규 노드 투입(신규 노드는 바로 서비스 가능 상태)



3. Operation Tools and Tips

3.1 Coordi-Reweight

노드 증설 작업 후 Reallocation 작업은 부하를 많이 유발하며, 서비스 IO 에도 영향을 미치게 됨
IO 상태에 따라 증설 작업을 판단하여 서비스에 영향을 최소화시키는 모듈임



12Bay 노드 단위 투입 시 3~6일 걸리며, 투입 중 노드 장애 대응이 어려움

crontab 등록 후 일정 단위씩 올릴 경우, slow request 발생 시 기민한 대응이 어려움

개선

Coordi-reweight

- Reweight 증설/감설 단위를 지정함
- Max backfilling pg 수 제한 → Client IO 영향도 감소
- Slow request 발생 시 중지/재실행
- 매일 지정된 시간에만 동작 → 밤시간에만 동작 등

3.2 Coordi-Split

노드 증설 후 PG Count 를 확인후 필요하다면 증가를 시켜야 함

PG Count 는 실제 Object 이동이 발생되므로, 서비스 IO 에 영향을 최소화 시키는 방안이 필요함

$$\text{PG Count} = \frac{(\text{Target PGs per OSD}) \times (\text{OSD \#}) \times (\%\text{Data})}{(\text{Size})}$$

Target PGs per OSD	This value should be populated based on the following guidance:
100	If the cluster OSD count is not expected to increase in the foreseeable future.
200	If the cluster OSD count is expected to increase (up to double the size) in the foreseeable future.

Doubling the PGs (e.g. from 8192 to 16384)

$ ceph osd <pool> pg_num <int>

$ ceph osd <pool> pgp_num <int>

- pg_num : splitting
- pgp_num : rebalancing / backfilling

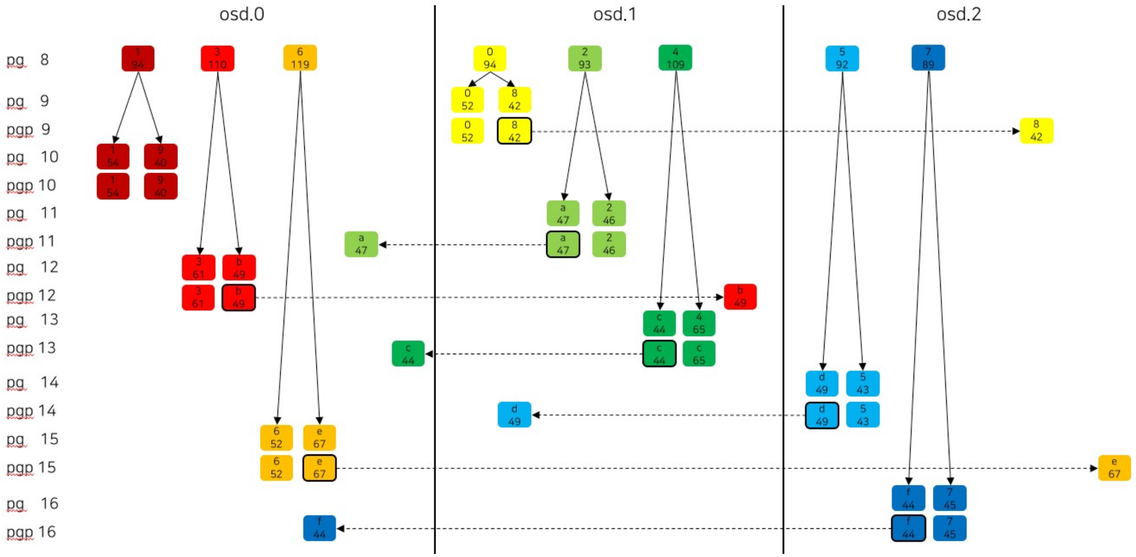
but there are several warnings in blogs or mailings lists, especially for production environments. Doubling the PGs (e.g. from 1024 to 2048) may bring down your cluster for some minutes, because creating, activating and peering the new PGs may strongly influence your client's traffic.

3.2 Coordi-Split

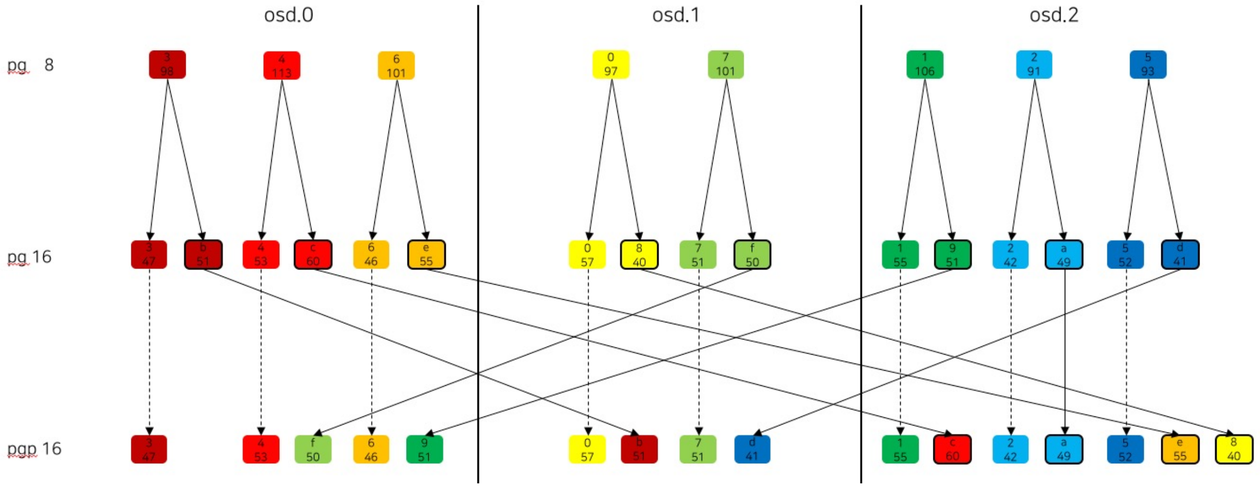
몇개 단위로 변경하는 것이 이동 횟수를 줄이는지에 대한 검토 결과,
128개 이상 단위로 변경하지 않는 한, 1개든, n 개든 비슷하게 이동됨

from PGs	to PGs	misplaced (%)	misplaced object	max backfill	max osd duration
323	324(+1)	0.10%	1620/1598703	1	-
324	326(+2)	0.21%	3294/1598703	2	0.24
326	330(+4)	0.41%	6612/1598703	4	0.34
330	338(+8)	0.86%	13806/1598703	6	0.67
338	354(+16)	1.63%	26003/1598703	9	2.12702
354	386(+32)	3.38%	53979/1598703	9	1.406066
386	450(+64)	6.49%	103677/1598703	18	2.09012
450	578(+128)	9.82%	156965/1598703	28	3.148092
578	834(+256)	13.25%	211857/1598703	45	4.038161

128이상 변경 시 3% misplace 덜 발생함



1개씩 8번의 pg 변경 시 7번의 이동 발생함



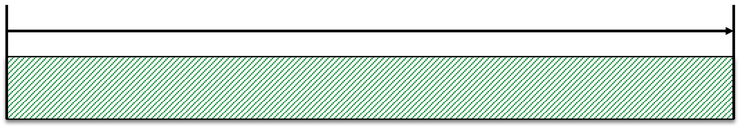
한번에 pg 8개 변경 시 7번의 이동 발생함

3.2 Coordi-Split

Ceph 의 Autoscaling PG 와 유사한 기능으로,

지정된 backfilling pg 수 이내에서 pg_num,pgp_num 을 일정 단위로 증가시키는 모듈임

Current pg : 1024 Target pg : 2048



한번에 올리게 되면 IO 가 일정시간 먹통 되는 문제가 발생함

crontab 등록 후 일정 단위씩 올릴 경우, slow request 발생 시 기민한 대응이 어려움

개선

Coordi-split

- split 증가 단위를 지정함
- Max backfilling pg 수 제한 → Client IO 영향도 감소
- Slow request 발생 시 중지/재실행
- 매일 지정된 시간에만 동작 → 밤시간에만 동작 등

3.3 bucket-pinner

CephFS 의 MDS Balancer 기능은 IO 지연을 유발시키게됨

Cephfs bucket (share) 을 특정 MDS 로 자동 pinning 시켜주는 모듈

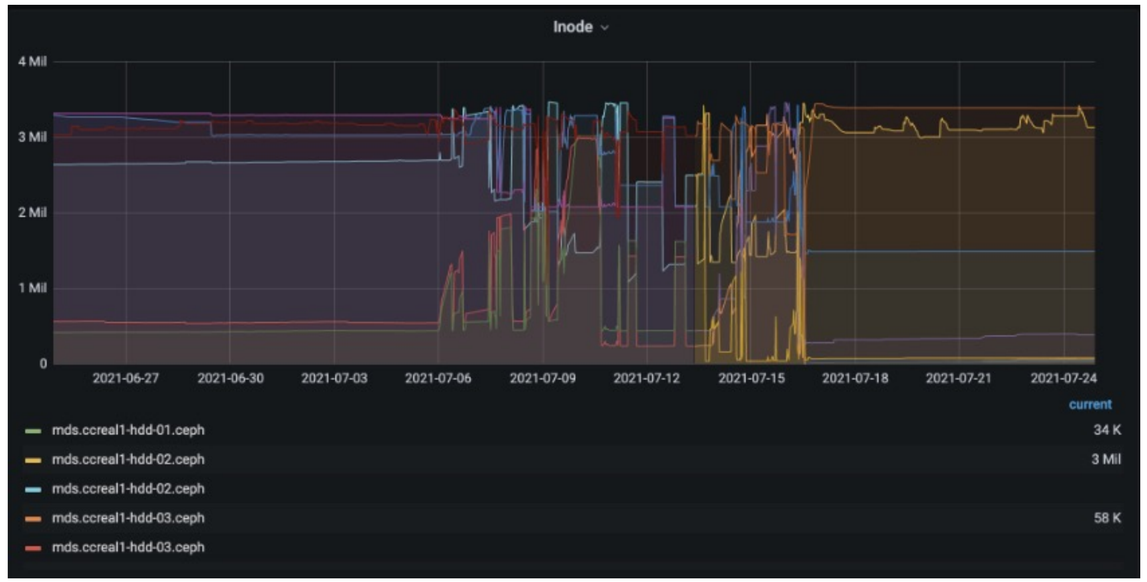
$$req\_rate = \frac{num\_requests}{the\_time\_span\_of\_the\_requests}$$

```
queue_len = messenger->get_dispatch_queue_len()
```

$$cpu\_load\_avg = \frac{(utime+stime)-(last\ get\ load\ utime+last\ get\ load\ stime)}{now-last\ get\ load\ time}$$

```
double mds_load_t::mds_load()
{
  switch(g_conf->mds_bal_mode) {
  case 0:
    return .8 * auth.meta_load() + .2 * all.meta_load() + req_rate + 10.0 * queue_len;
  case 1:
    return req_rate + 10.0*queue_len;
  case 2:
    return cpu_load_avg;
  }
  ceph_abort();
  return 0;
}
```

mds_bal_mode : 0 hybrid, 1 request, 2 cpu load

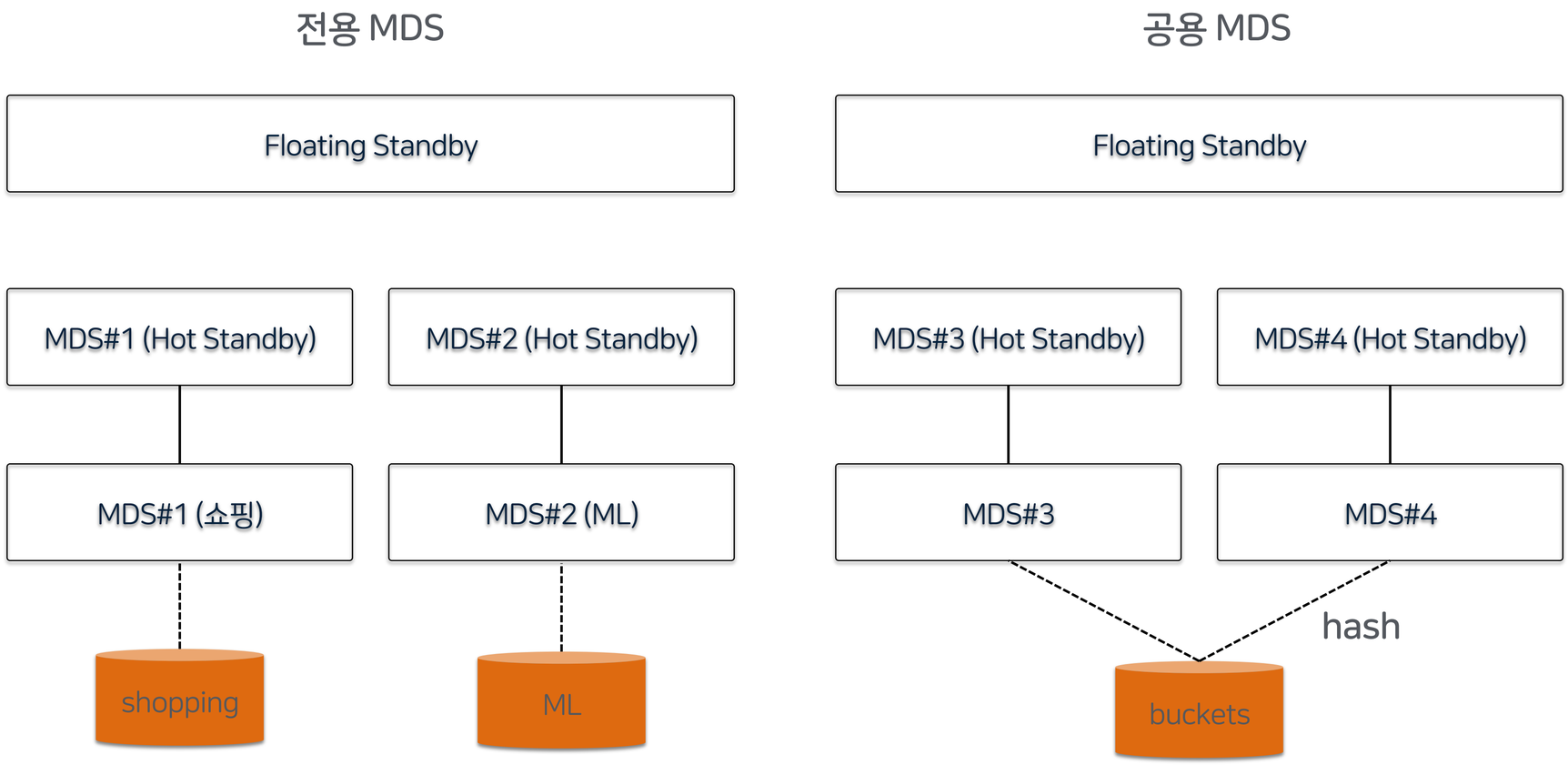


mds 간 지속적인 migration 으로 인해 io 지연발생

3.3 bucket-pinner

MDS 는 전용 MDS (높은 IO) 와 공용 MDS 로 구성되며, hot standby / floating standby 구성을 통해 가용성 확보

Subtree pinning 을 통해 bucket 과 mds 를 고정시키며, 이는 bucket-pinner 로 쉽게 적용함



3.4 Performance Monitoring

모든 이벤트는 사내 메신저를 통해 전달받고 있으며, 중요 알람 확인을 위해 레벨 단위로 처리

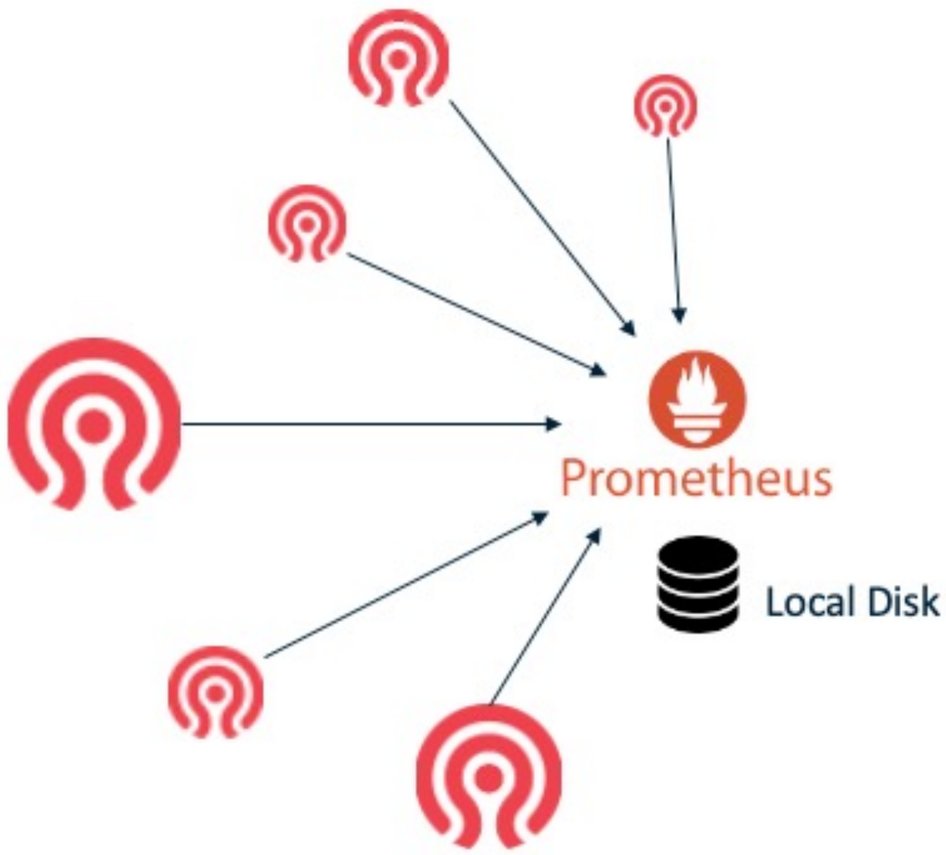
Prometheus – Grafana를 활용하여 모니터링 중이며, 다수 Exporter를 개발하여 사용

사내 메신저 연동을 위해 Grafana webhook서버를 개발하여 운영

3.4 Performance Monitoring

서비스 초기 중앙의 프로메테우스 서버로 모든 클러스터의 성능 데이터를 수집

메트릭이 많아지면서 중앙 수집 구조의 문제점들이 발생하기 시작함



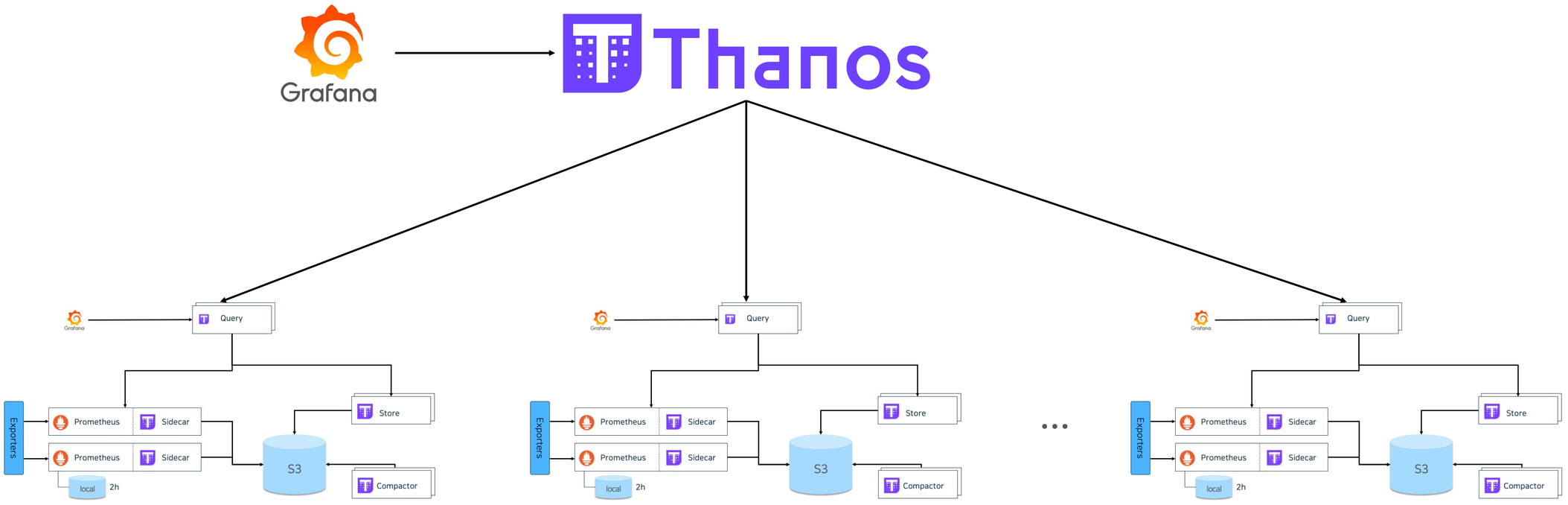
확장성 문제
- 로컬 디스크 부족 시 대응이 어려움

가용성 문제
- HA 지원되지 않음 -> 2개 기동

3.4 Performance Monitoring

모든 클러스터 정보를 한눈에 확인하기 위하여 글로벌 타노스를 적용함

상세 모니터링은 개별 로컬 클러스터를 통해 확인함



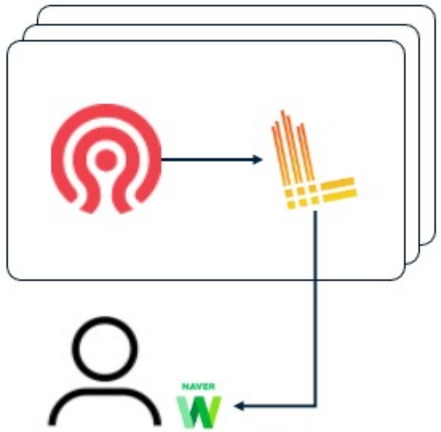
로컬 클러스터#1 로컬 클러스터#2 로컬 클러스터#N

3.5 Log Monitoring

글로벌 구축이 시작되면서 IDC간 로그 전송이 발생함

로그는 발생된 IDC에서 처리 후 Alert Event만 전송되는 구조로 변경함

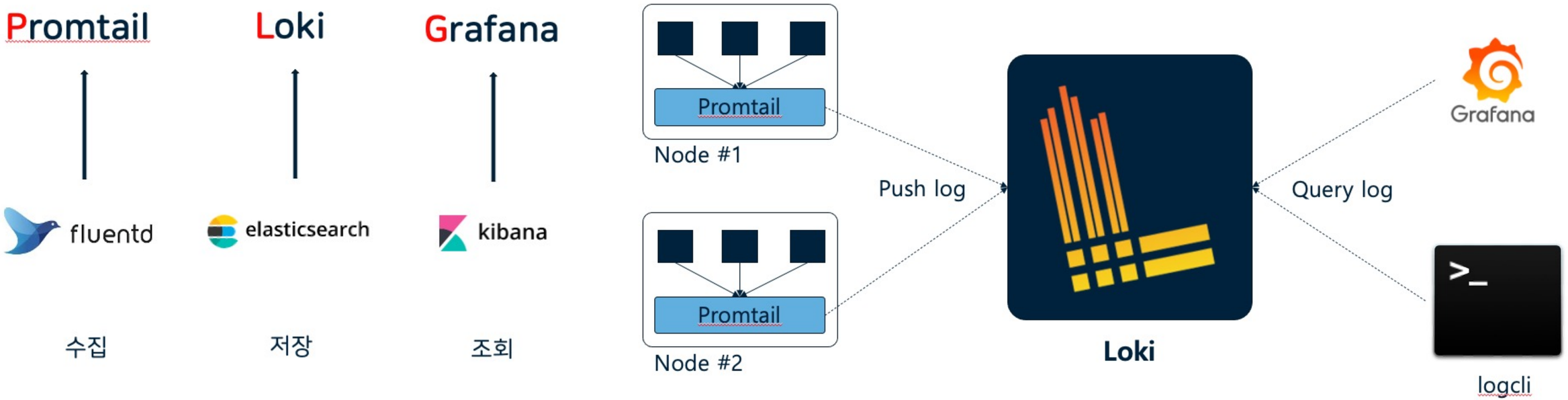
중앙수집구조



클러스터별 수집구조

3.5 Log Monitoring

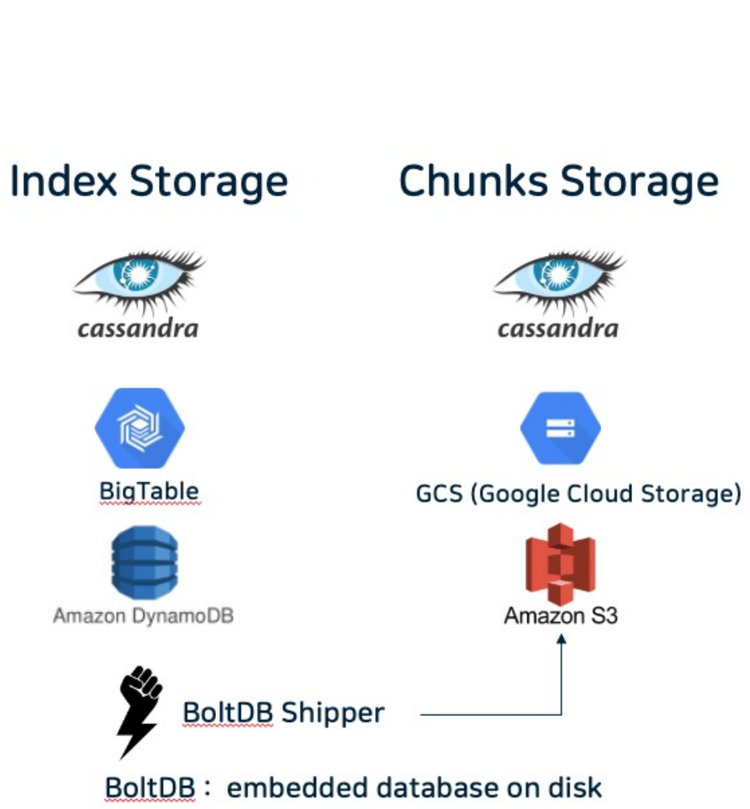
PLG Stack을 도입하고, 클러스터별 로그는 수집하는 구조로 변경

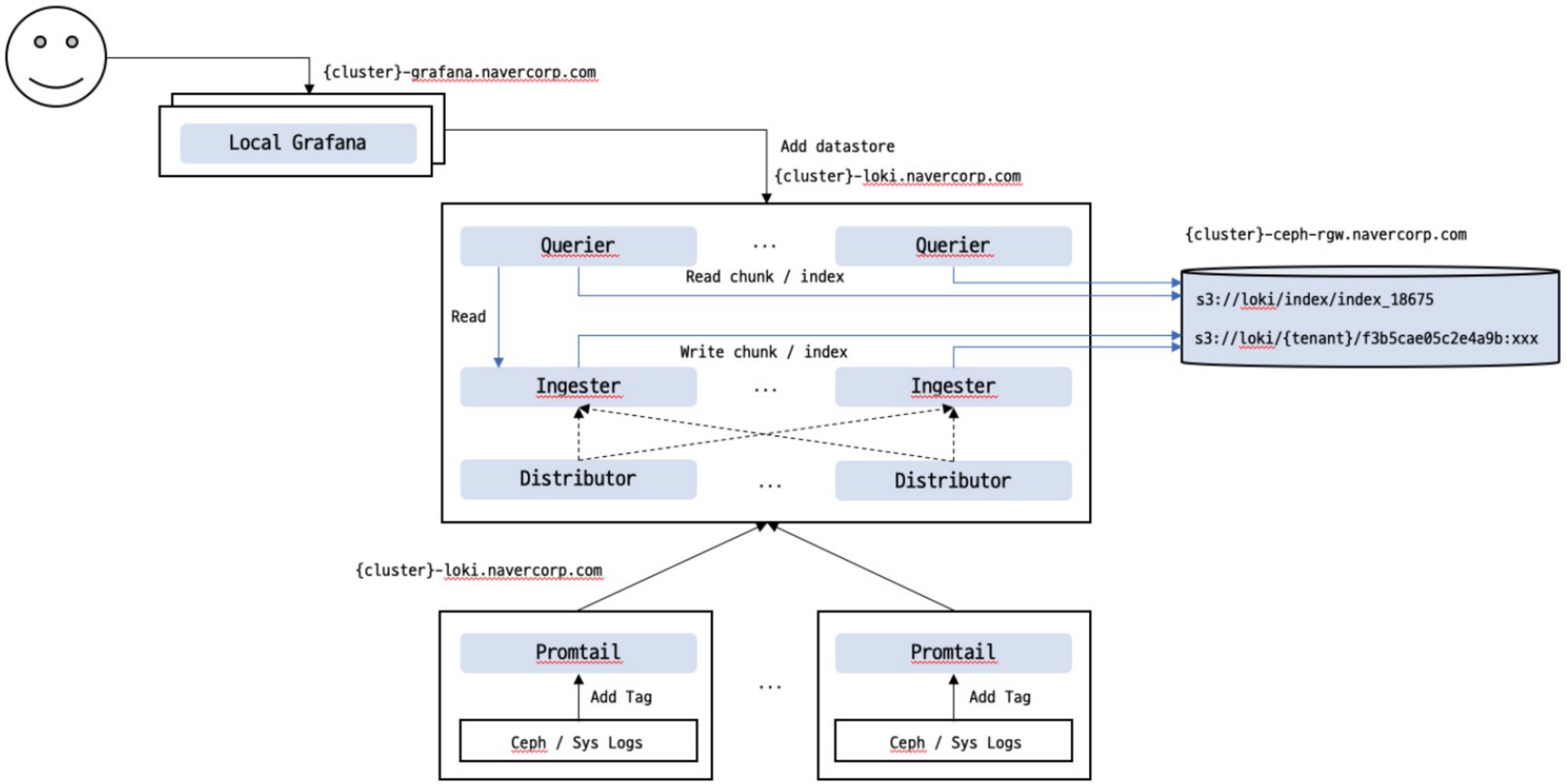


3.5 Log Monitoring

BoltDB와 BoltDB Shipper지원으로 S3만 있으면 Loki구축이 가능해짐

Ceph Object Storage S3 API를 통해 서비스 구축





Thank You

유장선 / jangseon.ryu@navercorp.com