Lessons learned from deploying SSD in NAVER services

NVRAMOS 2014

2014-10-30

Naver Labs

김태웅

NAVER | LABS

목차


NAVER
LABS

https://www.facebook.com/naverlabs

역할

- 네이버의 핵심기술의 연구/개발을 통해 기술을 내재화
- 학생 및 개발자들을 대상으로 기술과 꿈을 나누는 조직

개발자 컨퍼런스 DEVIEW

http://deview.kr/2014/main

목차

- Naver Labs
- **네이버의 데이터 저장 플랫폼**
- 네이버 서비스에서 SSD 활용
- 결론

The era of data explosion

매 1분마다

유튜브는 72시간 분량의 신규 동영상 업로드

이메일 사용자는 2억개 이상의 메일 발송

구글은 4M개 검색 질의 처리

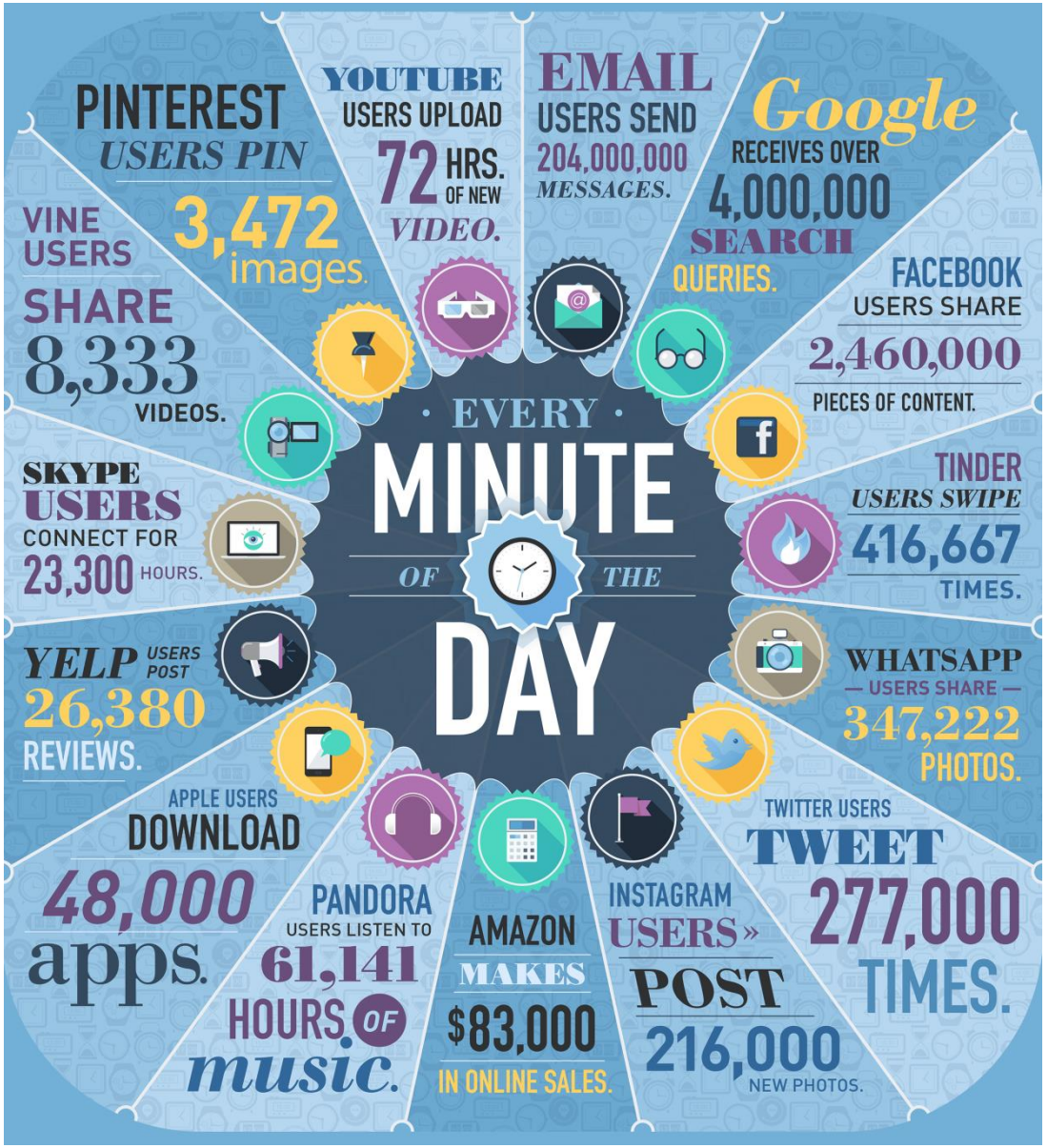
페이스북 사용자는 약 2.5M개의 컨텐츠 공유

트위터는 약 280K개의 트윗

인스타그램은 약 220K개의 사진 업로드

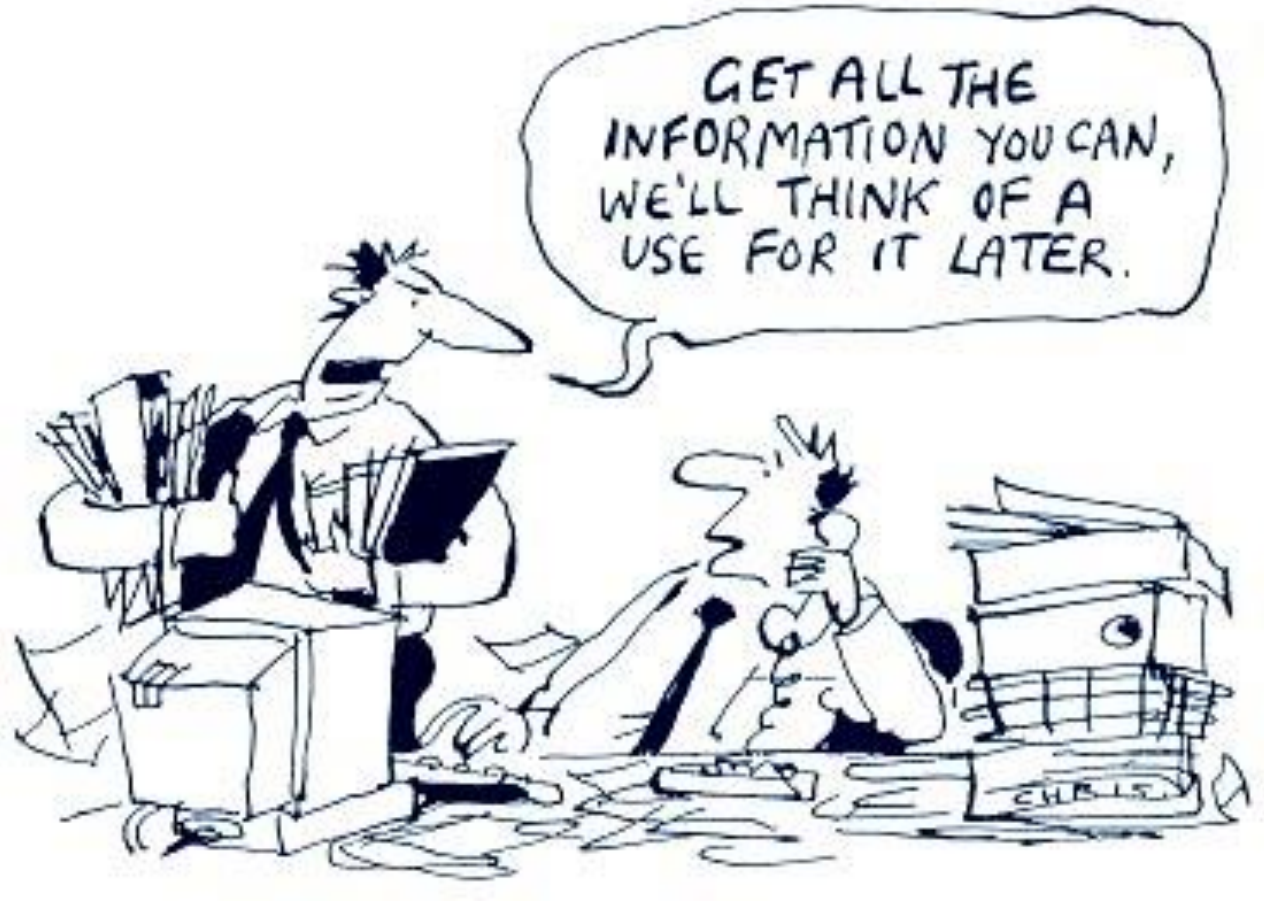
하루 최대 130억건의 communication
분당 약 9M건 이상 메시지, 동영상, 사진 송수신

http://linecorp.com/en/pr/news/en/2014/845



출처: http://newstex.com/2014/07/12/the-data-explosion-in-2014-minute-by-minute-infographic/

이용자에 의해 생성되는 데이터도 있지만



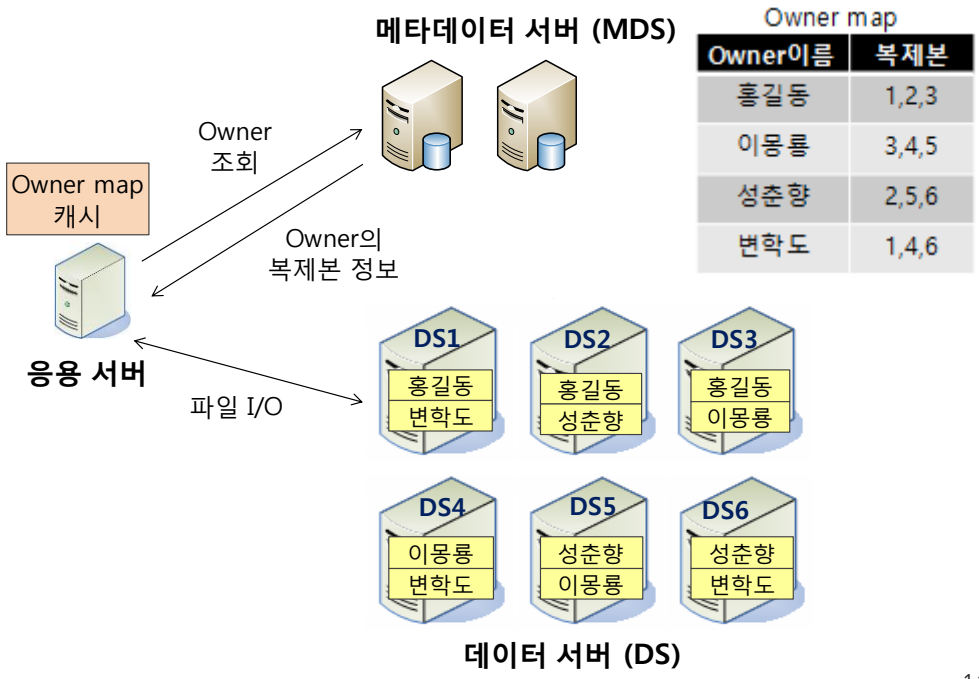
출처: http://www.pinterest.com/pin/18929260905459702/

네이버의 데이터 저장 플랫폼

- 파일 저장소
  - 분산 파일시스템: OwFS, Papyrus
- 데이터베이스
  - DBMS: CUBRID
  - 분산 DB 클러스터링: nBase-T
- 분산 Key/value 저장소
  - nBase-ARC
- 메모리 캐시
  - ARCUS

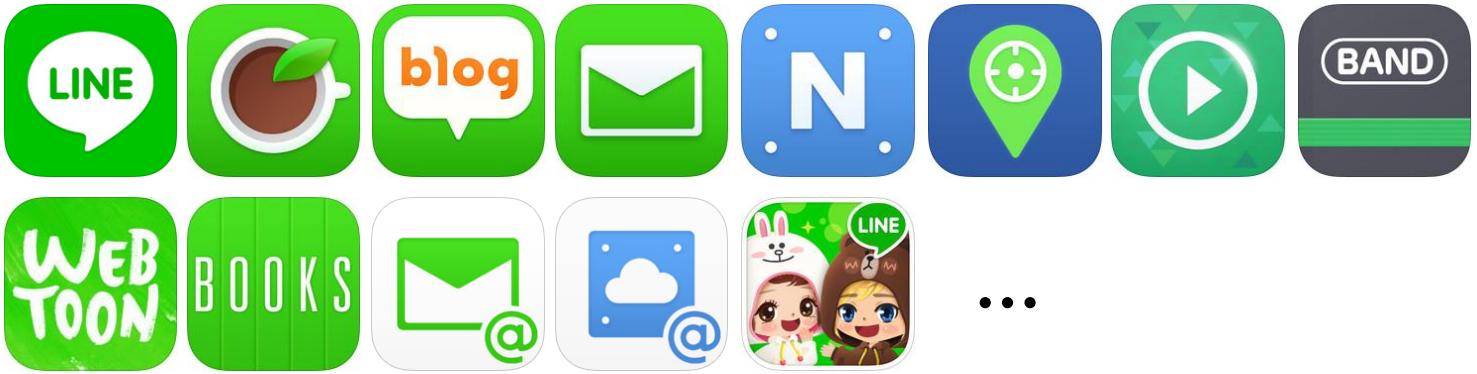
OwFS

- What is OwFS? **Ow**ner-based **F**ile **S**ystem
  - 네이버의 포털 서비스 환경에 적합하도록 개발한 대규모 분산 파일시스템으로,
  - **여러 서버들의 디스크 공간들을 묶어 하나의 대규모 파일 저장 공간을 생성**
  - Owner
    - 서로 관련 있는 파일들을 묶어서 저장하는 단위 (container 개념)
    - 수많은 owner의 합이 전체 파일 시스템을 구성
    - Owner 별로 3개의 복제본을 서로 다른 서버에 저장
  - OwFS의 특징
    - 가용성
    - 확장성
    - 성능
    - 운영 편의성
    - 점검시 서비스 downtime이 없음



OwFS

- 적용 서비스
  - 네이버/LINE/NHN Entertainment의 200개 이상 서비스

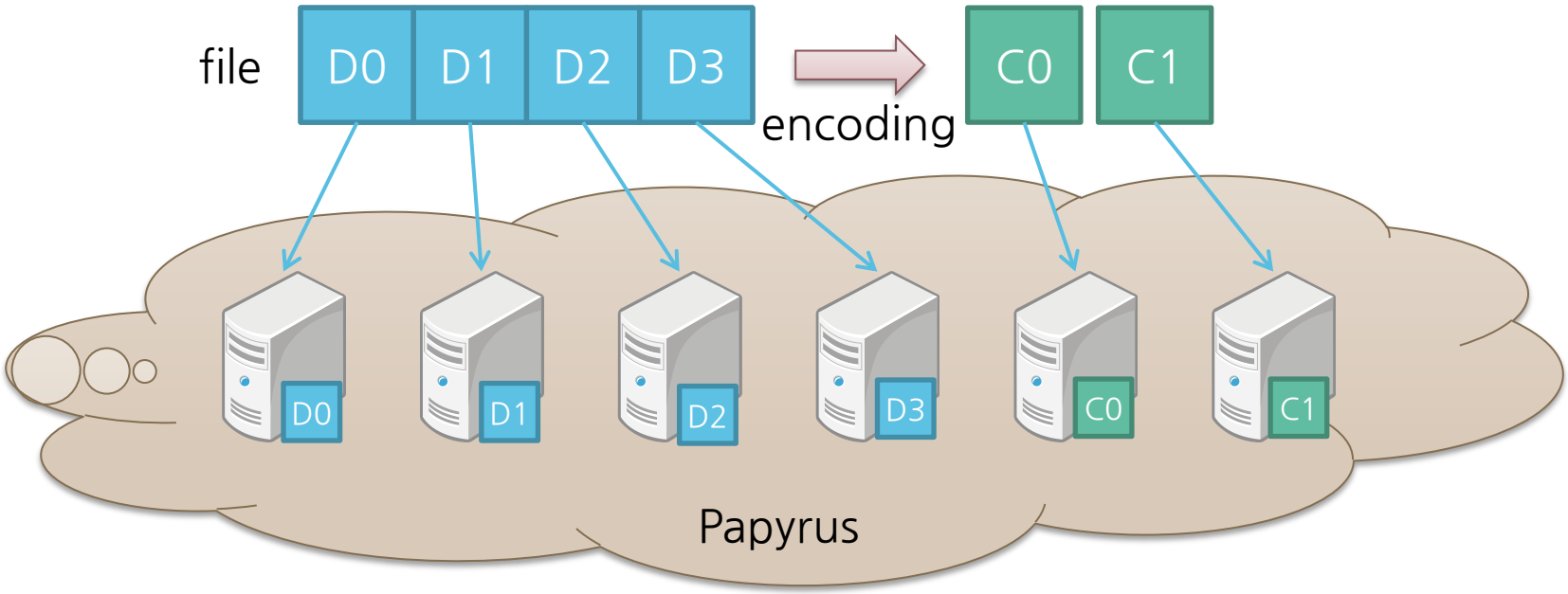


- 일반적으로 데이터 저장서버는 대용량의 SATA 디스크 기반으로 구축
- 고성능을 위해서 최신 데이터는 SSD에 우선 저장하고 일정기간후 HDD로 이동시키는 tiering을 적용하기도 함

- 범용 파일 저장 목적외에
  - ViSTO (Virtual iSCSI Storage over OwFS): Amazon EBS와 유사하게 네이버 내부 개발자 클라우드의 VM환경에서 동적으로 블록 스토리지를 제공

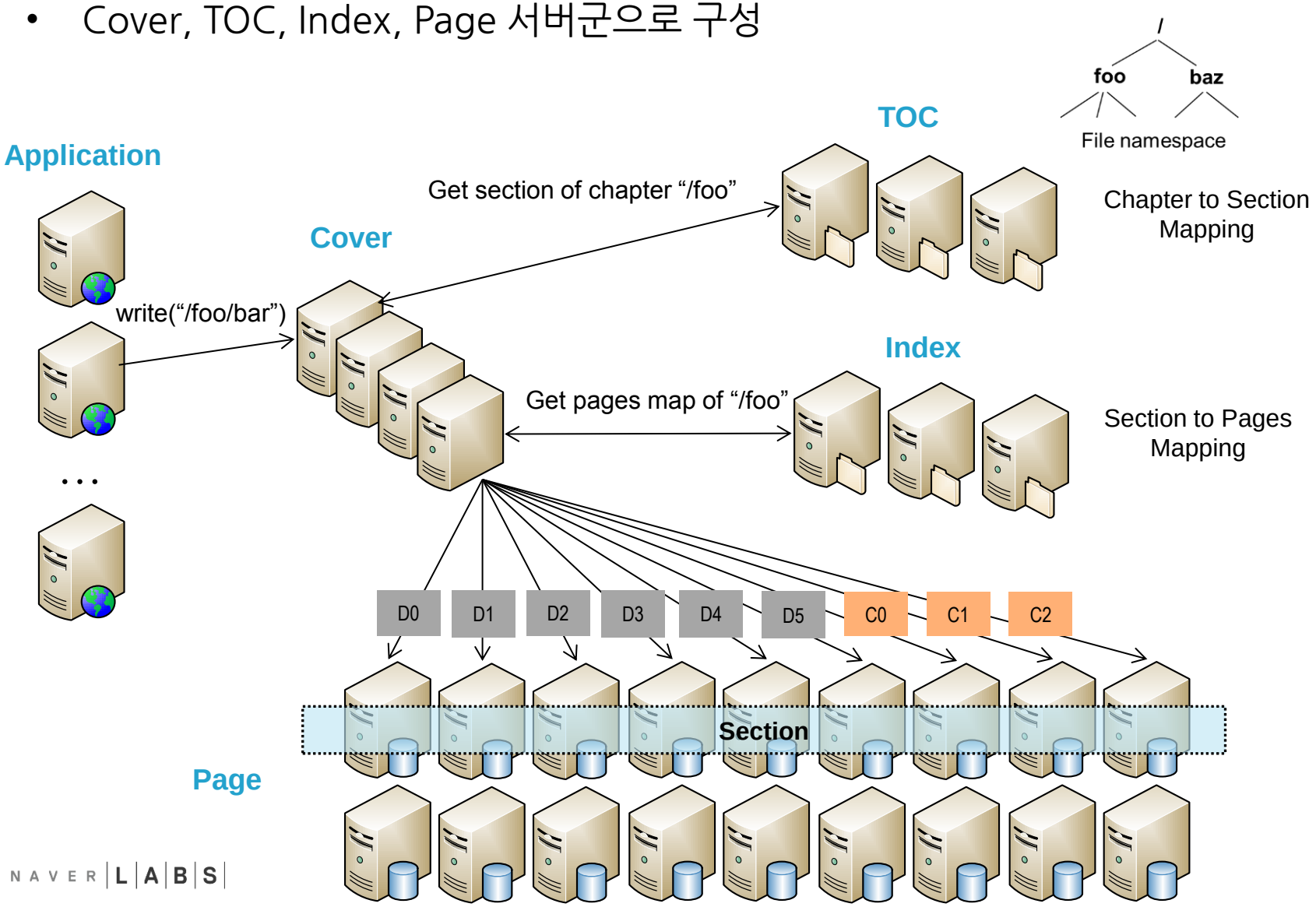
Papyrus

- 분산 파일 시스템의 Storage tiering에서 secondary storage
- 저장 비용이 효율이 좋은 파일 archive용 스토리지
- 특징
  - OwFS와 같은 복제본 기반의 데이터 가용성을 사용하지 않고 Erasure coding을 사용한 데이터 저장 방식 사용



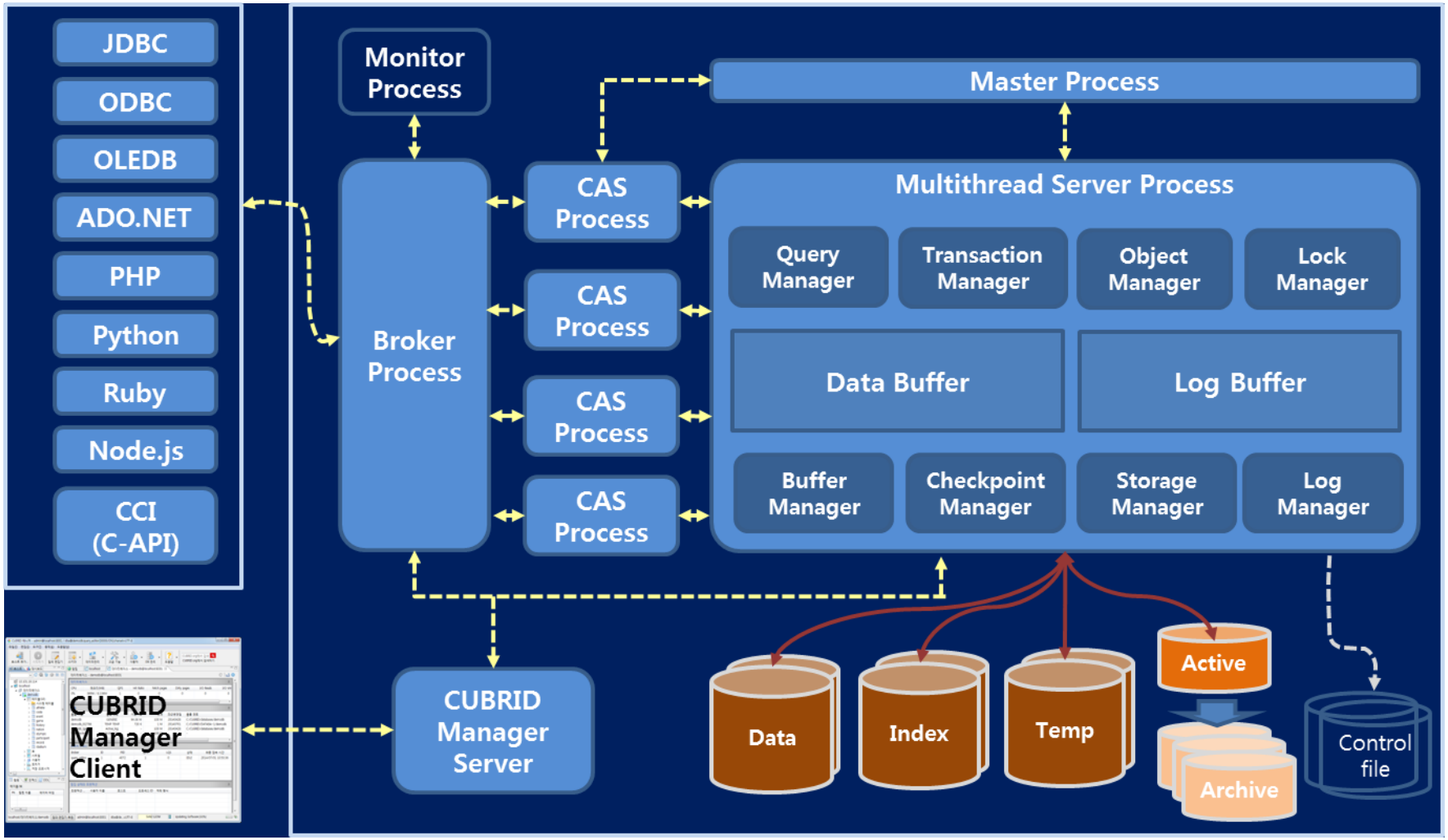
Papyrus architecture

- Cover, TOC, Index, Page 서버군으로 구성



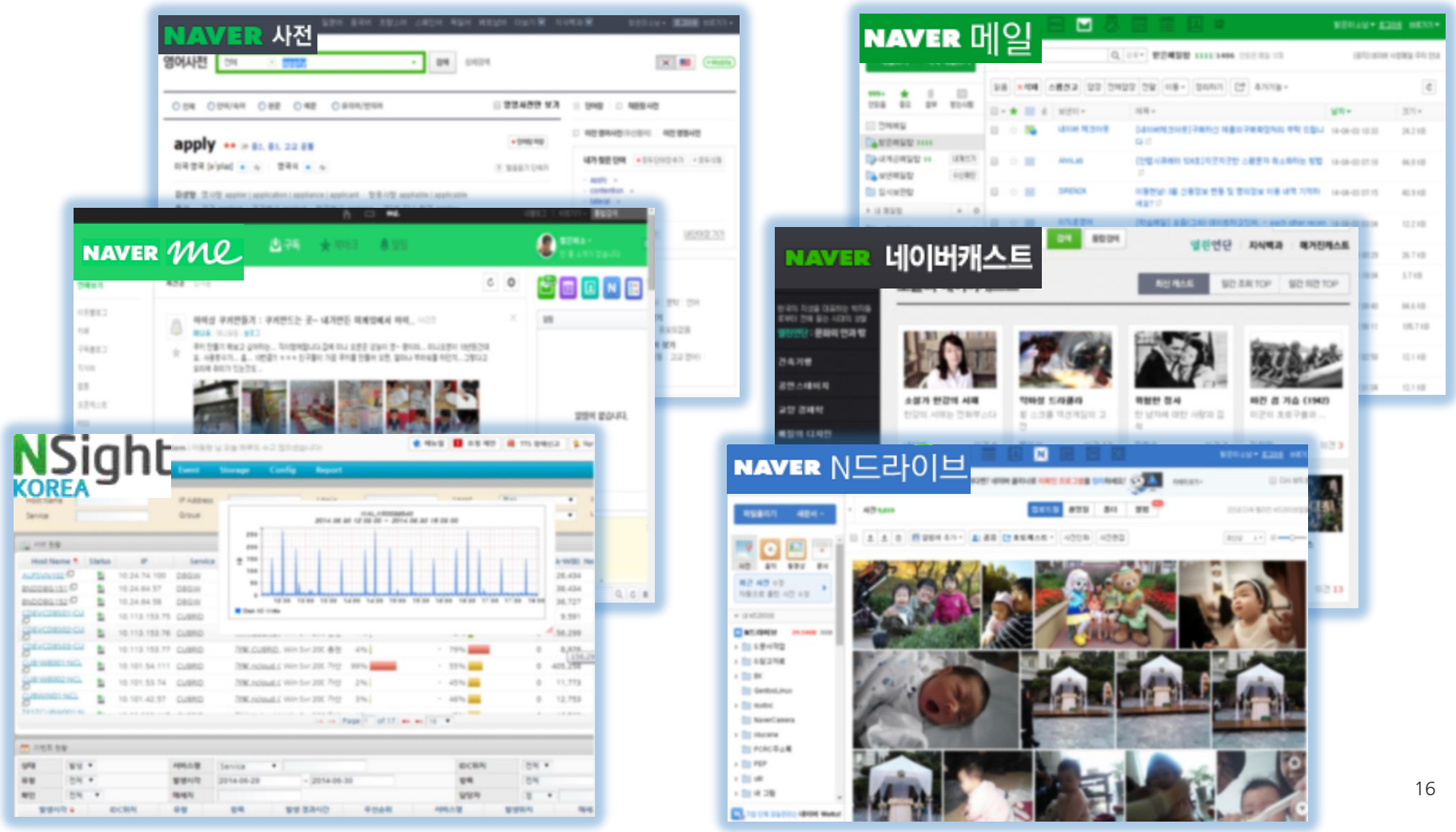
CUBRID DBMS

네이버의 오픈소스 DBMS http://www.cubrid.org



CUBRID 적용 서비스

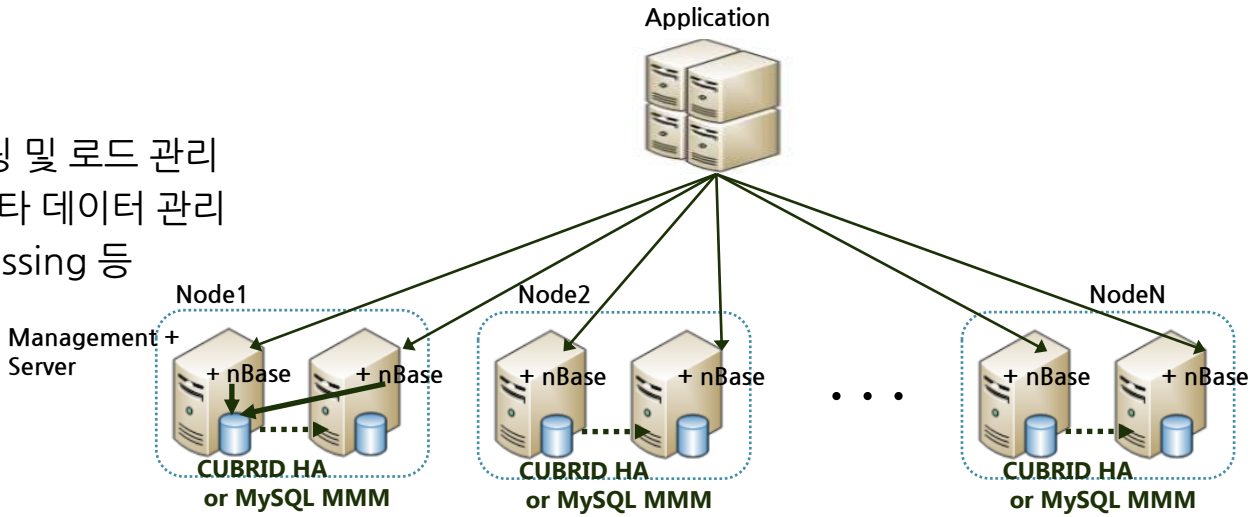
- 메일, 사전, Naver me, 네이버 캐스트, N드라이브 등 NAVER 주요 서비스에 적용
- 사내 시스템 모니터링 사이트인 Nsight 등 사내 주요 서비스에 적용



nBase-T

- **nBase-T**
  - 분산과 확장성을 제공하는 DB middleware
  - 분산 키 별로 container 라는 개별 공간을 제공. container 별로 RDBMS 기능을 지원
  - Online data migration을 통한 노드 증설/감설/물리이전/밸런싱 기능을 제공

- **주요 기능**
  - Online data migration
  - 병렬 작업에 대한 스케쥴링 및 로드 관리
  - Membership 상태 및 메타 데이터 관리
  - Distributed query processing 등



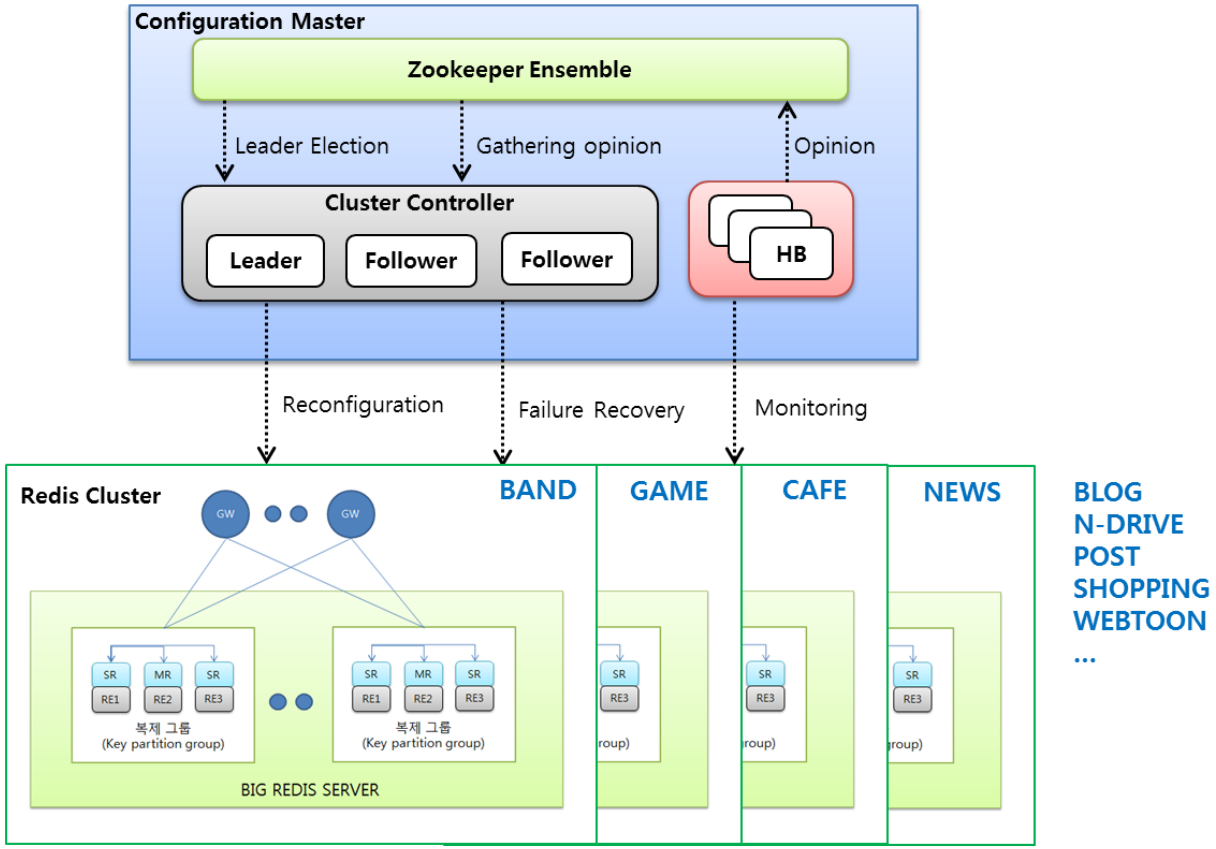
- **적용 서비스**
  - 네이버 : 메일, 블로그, …
  - 라인: 분산 코인 시스템, 세션/인증, 게임DB,…
  - 기타 라인플레이 게임DB, NHN ENT 게임메일/게임DB

nBase-ARC

Redis 기반의 분산 메모리 저장소로 다양한 구조체에 대한 in-memory 고속 연산을 분산 환경에서 동일하게 제공

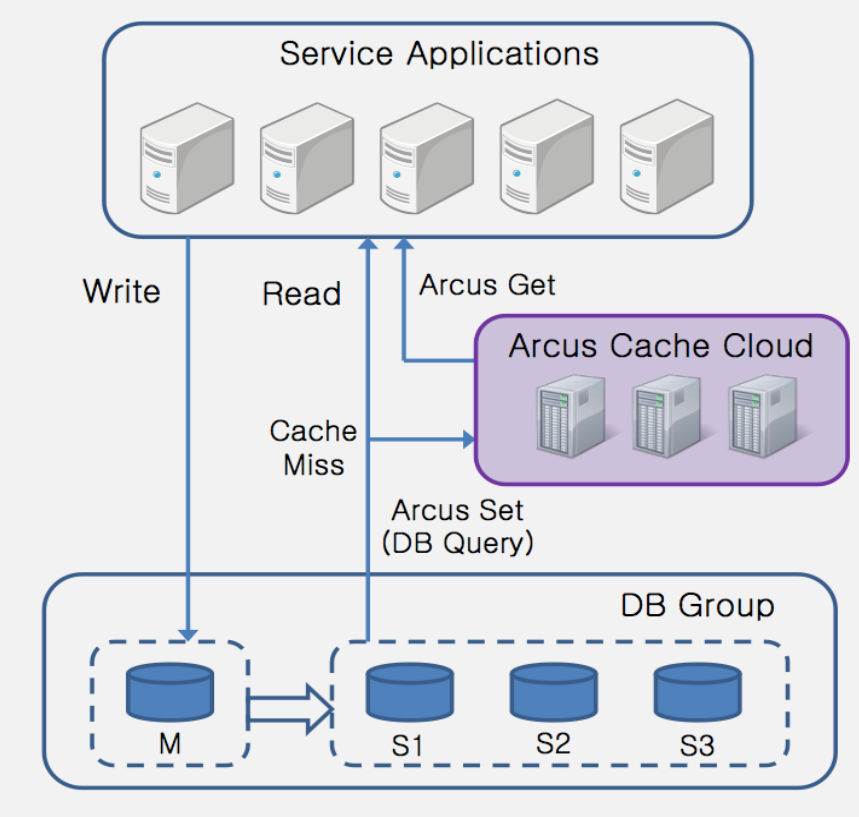
고가용 Multi-Cluster service pool

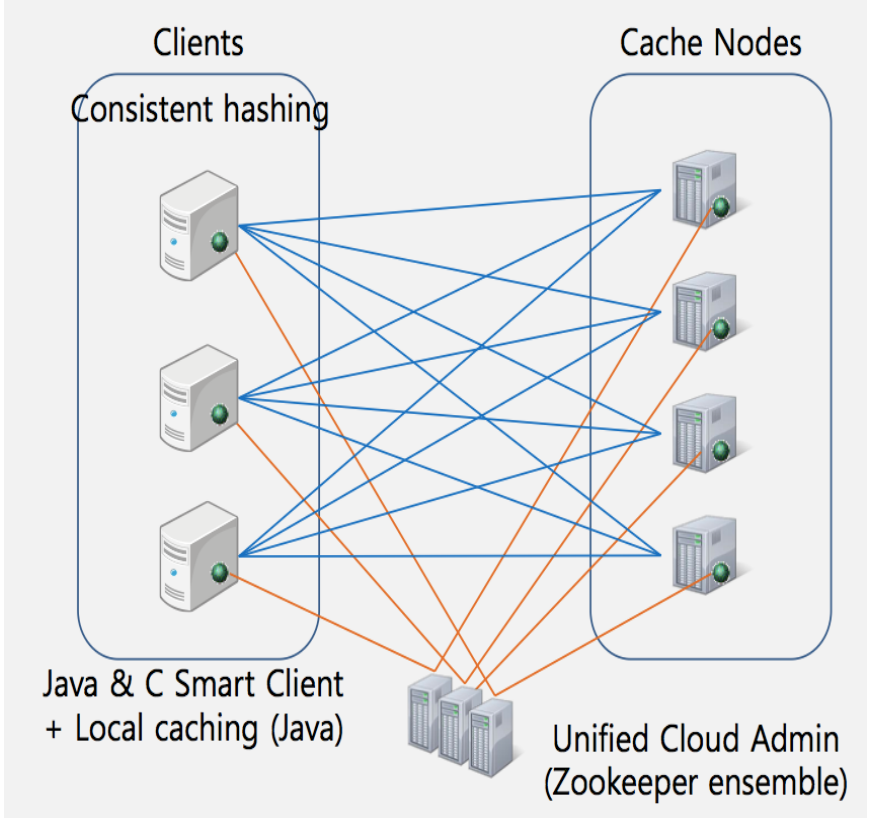
일관성을 보장하는 복제 layer 기반의 복제그룹의 집합이며, nBase-ARC는 복수의 cluster를 운용



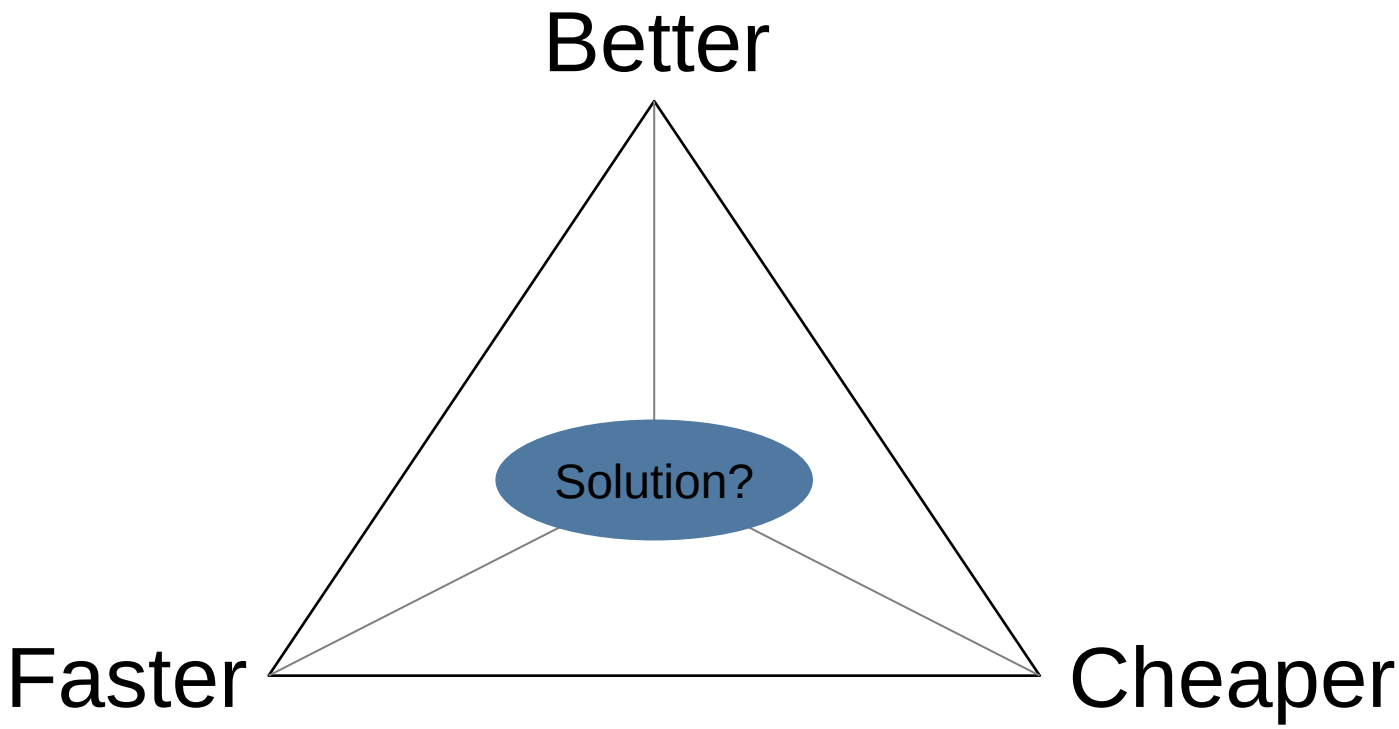
페타바이트급 대용량 이기종 클러스터드 DBMS SW개발 (과제번호 10041311)

ARCUS (메모리캐시 클라우드)





서비스와 플랫폼 개발 및 구축시 고려사항



- 데이터 스토리지 관점에서
  - **Better** 데이터의 durability를 높이려면 더 많은 복제본을 동기적으로 기록
  - **Faster** 데이터 IO의 latency를 줄이려면 메모리와 같은 고속매체 의존성이 높아짐
  - **Cheaper** 비용만 생각한다면 성능과 품질의 희생이 필요

목차

- Naver Labs
- 네이버의 데이터 저장 플랫폼
- **네이버 서비스에서 SSD 활용**
- 결론

A brief retrospect

- 네이버는 2010년부터 SSD 도입 시작
  - 고성능 DB서버에 SSD 장착, 높은 IOPS를 요구하는 서비스에 시범 적용
  - SSD에 대한 막연한 불안감
    - 수명과 안정성을 고려하여 SLC 모델 위주 도입
    - GC로 인한 성능저하를 우려하여 용량 기준을 85%만 사용하는 것을 기준

- 2011년
  - DBMS는 대부분 SAS 디스크를 사용
  - 읽기 부하가 높은 DB 서비스는 메모리 캐시 + DBMS 조합으로 구성
  - 쓰기 부하가 높은 DB에만 SSD 장착

- 2012년 ~
  - DB 쓰기 부하가 높은 응용의 요구사항 증가
  - SSD의 구축 비중이 높아짐

SSD도입시 고려했던 사항

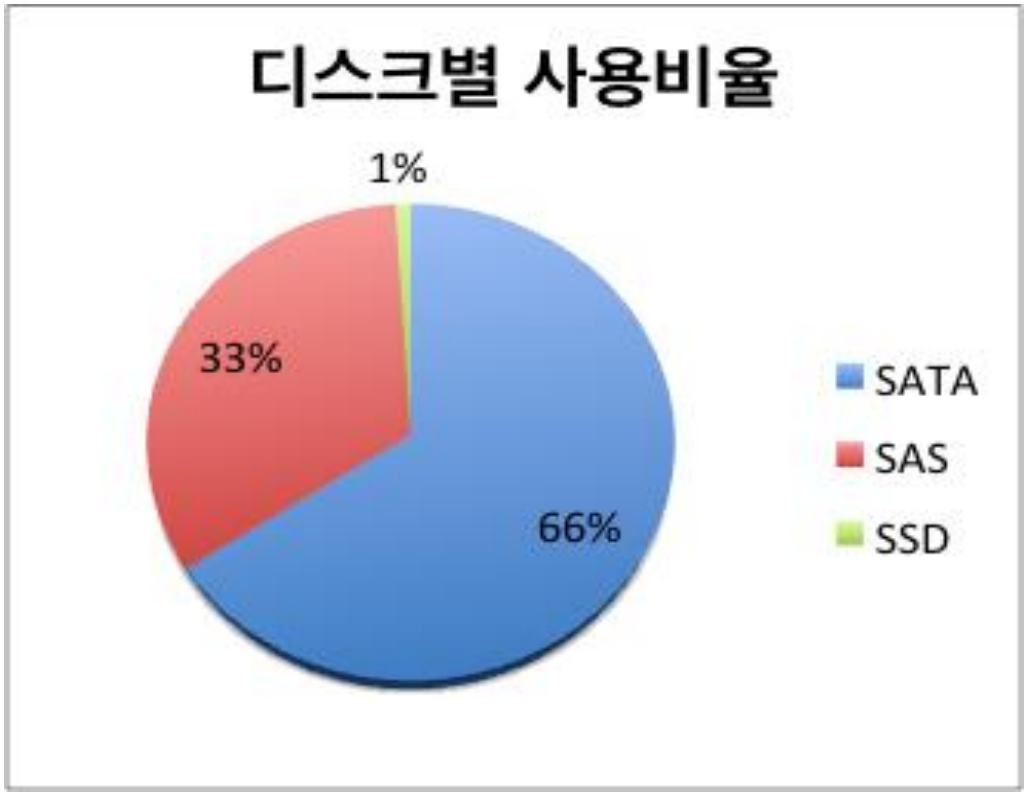
- SSD의 장점
  - 성능관점: HDD에 비해 우수한 성능 (random I/O)
  - IDC 운영관점: 소음과 발열이 적고 소비전력과 상면공간 절약

- SSD의 단점
  - 높은 비용과 소용량 (vs. HDD)
  - 쓰기 회수 제한 (데이터의 성격에 따라 다름)
    - Write Once Read Many 성격의 사진, 뮤직, 동영상 서비스에는 우수
    - 데이터베이스와 같이 데이터의 삽입, 삭제, 변경이 많은 경우는 내구성 문제

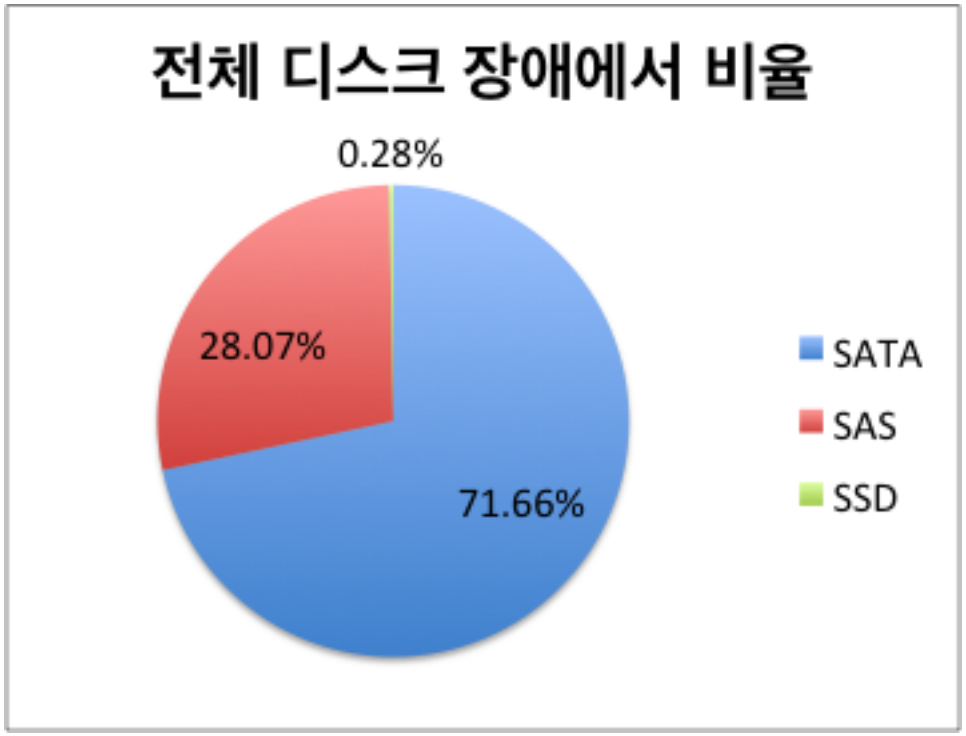
- 스토리지 구축시 HDD/SSD 선택기준
  - 요청 처리량, 목표 응답시간, 워크로드 특성 (읽기:쓰기 비율)
  - 성능 요구사항에 부합되는 비용 효율성 고려
    - DB 서버의 scale up, scale out시 비용
    - 메모리 캐시 클라우드의 활용시 비용

SSD 사용 현황

- SSD는 어떤 용도로 주로 사용되는가?
  - 고성능 데이터베이스 서버
  - 고성능 파일 서비스를 위해 SSD-HDD로 Tiering



SSD의 고장률 통계



- Type별 장애 비율

Type	장애 비율
SATA	1.57%
SAS	1.21%
SSD	0.41%

SSD 고장이 발생시 RAID로 구성된 다른 SSD도 고장날 가능성이 높다는 것을 경험하였고 SSD의 수명을 예측하고 대비하는 것이 매우 중요

PCIe Flash card

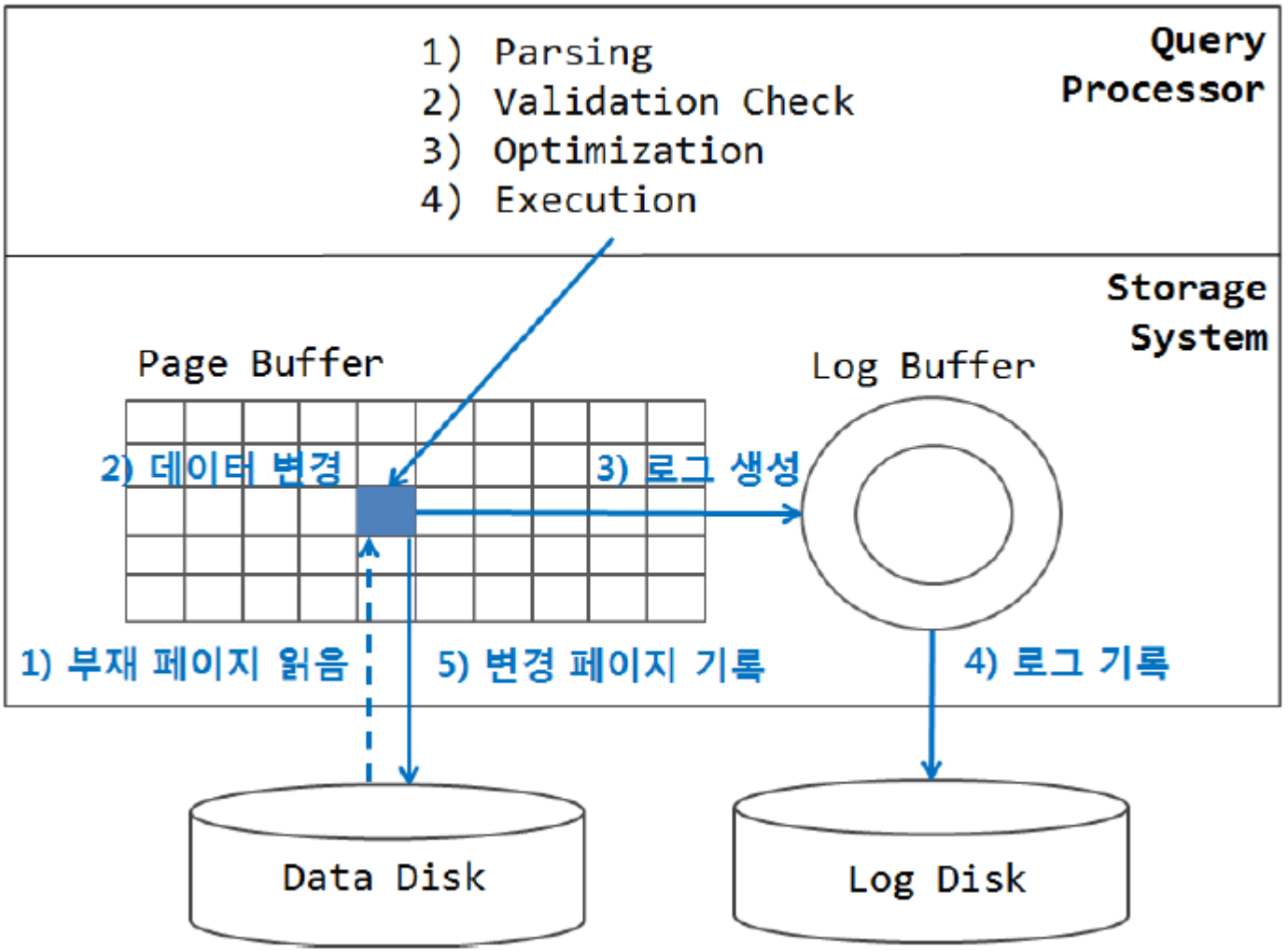
- Fusion-io의 ioDrive, ioScale
- 초고성능 DB 서버 구축시 주로 사용

- PCIe Flash Card의 경우 문제점
  - 운영관점: 저장장치 고장시 교체의 문제
  - 비용관점: SSD에 가격이 매우 비쌈

- DB와 I/O intensive 서버들은 상대적으로 CPU utilization이 낮으므로 이런 서버들을 consolidation하는 용도로도 PCIe Flash Card를 활용 가능

DBMS

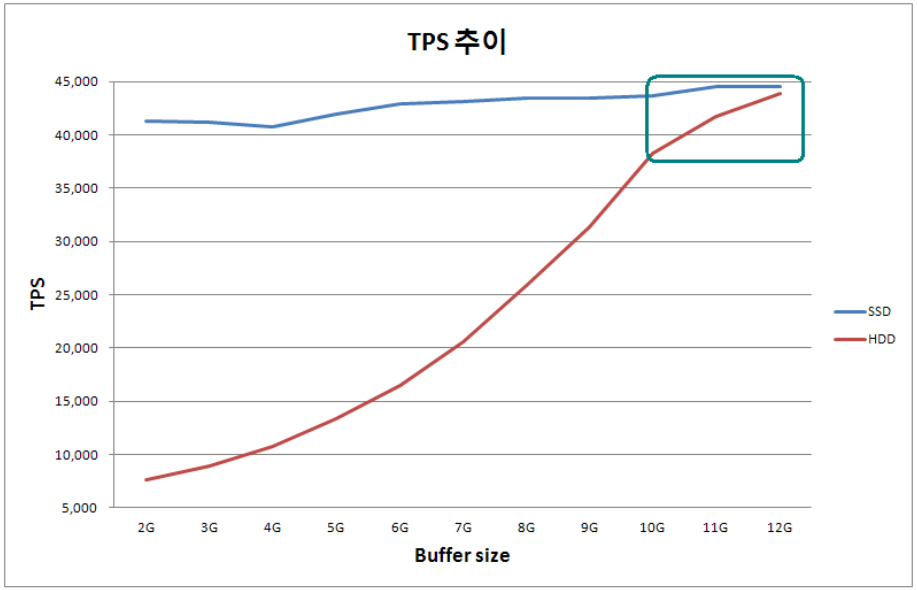
- Page buffer와 디스크간 I/O는 random I/O

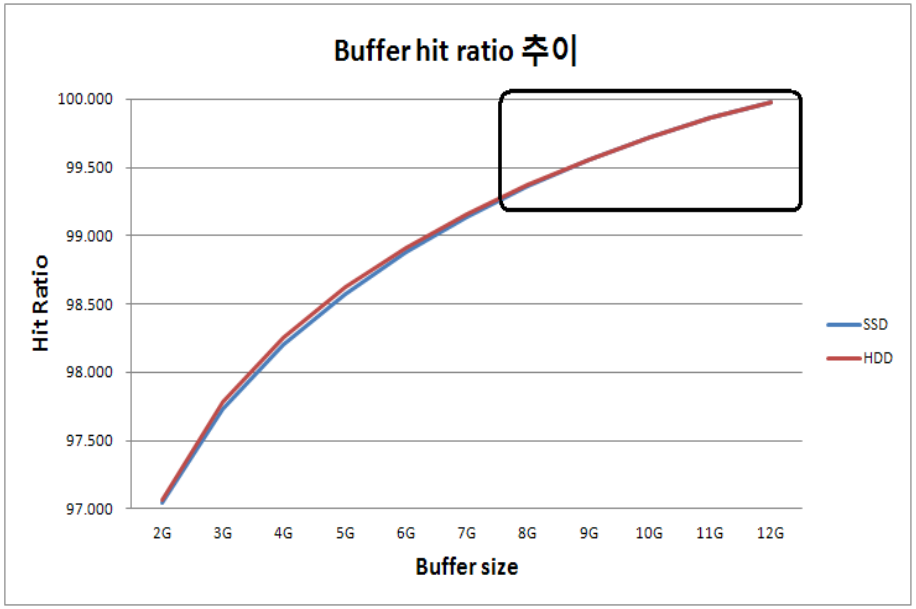


SSD 도입으로 인한 SELECT 성능 향상

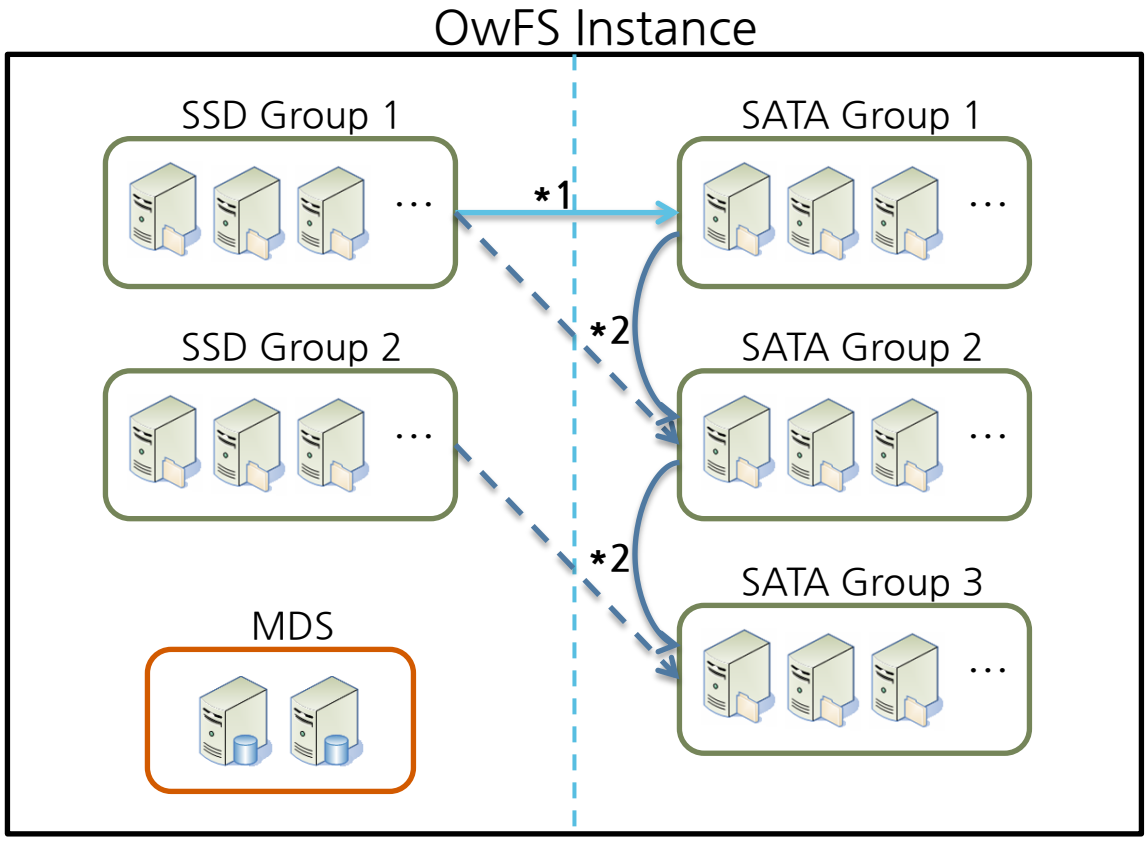
- YCSB를 이용한 성능측정
  - DB 데이터버퍼 크기와 저장매체간의 성능 변화 상관관계
  - CUBRID의 버퍼 크기를 변경하면서 SELECT TPS 측정

http://helloworld.naver.com/helloworld/7005





분산 파일시스템에서 Tiering



*1 파일 write 직후 async하게 복제
*2 일정 기간 경과 후 SSD Group의 복제본을 제거하고 SATA 그룹 내에서 3 copy 복제

Many small files 저장 문제

- 1GB 블록 디바이스를 각각 Ext4, XFS로 파일시스템을 만든 후 디렉토리당 1KB 크기의 파일을 10,000개씩 생성
  - 1GB는 약 1,000,000 KB이므로, 백만개에 근접한 개수의 파일을 저장하는 것을 기대

파일시스템	File system full	File system usage
Ext4	65,518개 파일 생성	31%
XFS	234,482개 파일 생성	100%

  - mkfs의 default 옵션: 파일시스템의 블록크기 4KB, ext4의 경우 1GB 파티션의 경우 inode 개수가 65,536개로 파일시스템이 만들어짐

- 파일시스템 Tuning

파일시스템	File system full	mkfs 명령 옵션
Ext4	796,702개 파일 생성	-b 1024 -N 810000
XFS	810,172개 파일 생성	-bsize=1024 -imaxpct=0

Many small files 저장문제 (계속)

- 파일 저장 서버
  - 12 디스크 bay, 4TB SATA 디스크
  - RAID5 구성으로 6개씩 구성하면 20TB 볼륨 2개
  - RAID5 구성으로 12개를 구성하면 44TB 볼륨 1개

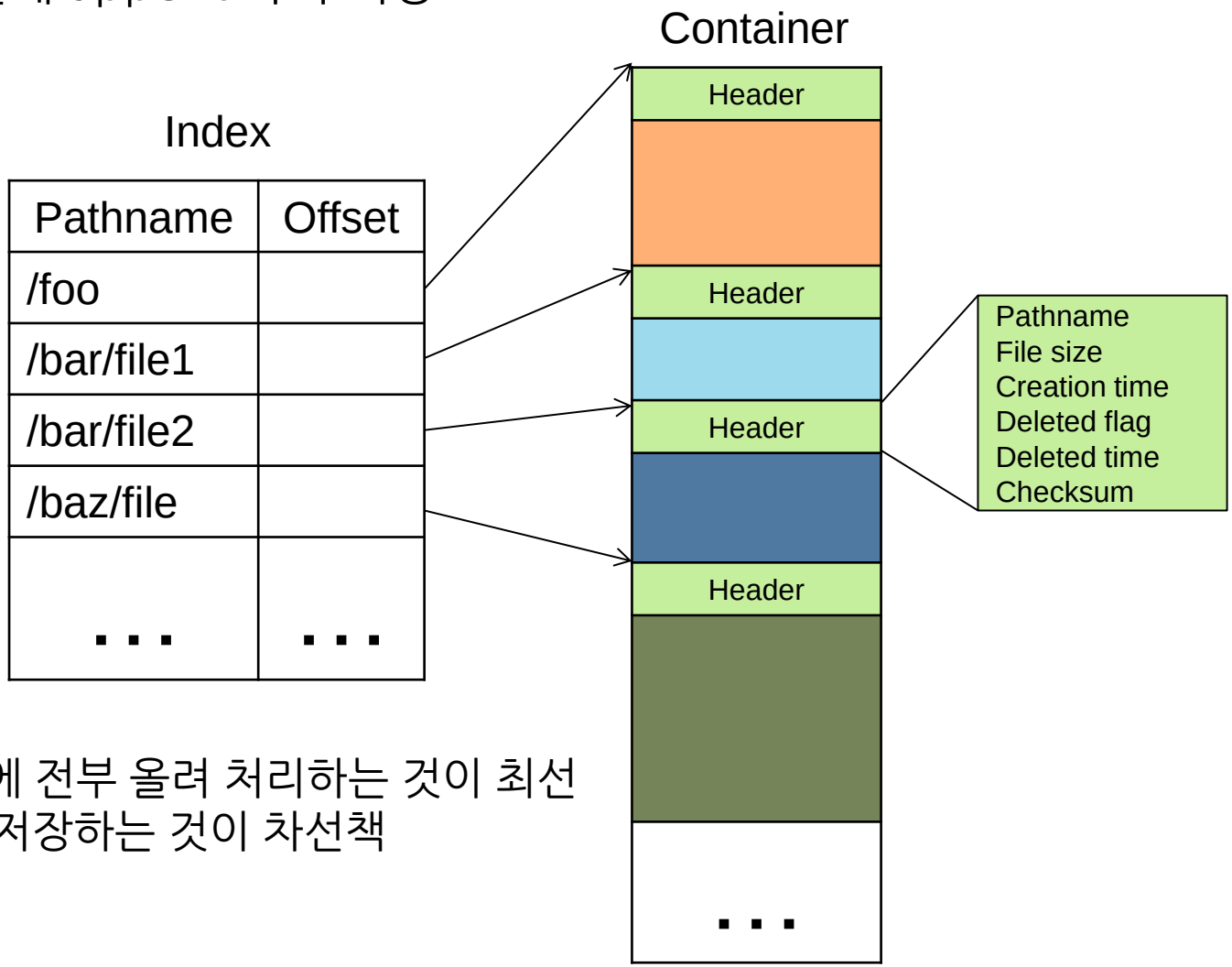
  - 4KB 미만의 가변크기의 작은 파일을 저장한다면,
    - 20TB 볼륨에는 최소 50억개 이상의 파일이 저장됨 (5,3 Billion)
    - 44TB 볼륨에는 최소 110억개 이상의 파일이 저장됨 (11.8 Billion)

  - 디렉토리 블록과 inode 블록을 메모리에 충분히 캐싱하기 어려움
  - 파일을 접근하려면 로컬 파일시스템에서 지속적인 random I/O 유발

따라서, Many small files 저장서버 구성시 HDD보다는 SSD가 유리
그러나, 문제는 비용

Many small files에 유리한 저장구조

Small 파일을 파일시스템상의 개별 파일로 저장하지 않고
Container 파일에 append하여 저장



Index를 메모리에 전부 올려 처리하는 것이 최선
Index를 SSD에 저장하는 것이 차선책

목차

- Naver Labs
- 네이버의 데이터 저장 플랫폼
- 네이버 서비스에서 SSD 활용
- **결론**

SSD에 대한 요구사항

- Random I/O 성능 극대화
  - 복제를 이용한 가용성 제공 및 고장복구
  - SSD에 RAM 버퍼를 두고 쓰기요청을 RAM 버퍼에 기록하고 반환해도 허용될 수 있는 여지가 있음

- Maximum read latency 최소화
  - Average read latency도 중요하지만 maximum read latency가 더 중요
  - SSD 내부에서 읽기 요청을 쓰기 요청보다 높은 우선순위로 처리하거나 GC의 수행주기등을 조절하여 읽기 요청의 응답시간 편차를 감소

- DBMS에서 단일 페이지의 atomic write
  - DBMS가 사용하는 페이지 크기와 SSD 내부의 페이지 크기가 일치하지 않으면, 단일 페이지 쓰기가 SSD 내부에서 복수의 페이지 쓰기가 될 수 있음
  - 이러한 mismatch에 대비하여 DBMS들은 보완책을 사용하게 되는데 SSD의 내부 페이지 크기를 응용이 요구하는 크기로 맞출 수 있다면 좋겠음

박준현, 신기빈, 송창현, “IDC에서의 애플리케이션 이슈와 SSD 요구 특성”,
전자공학회지(The Magazine of the IEEK) 제41권 5호, pp.57-67, 2014년 5월

SSD에 대한 요구사항

- SSD 내부동작 상태의 조회기능
  - SSD 내부의 페이지/블록 크기, GC 수행 시점과 주기, wear leveling, 에러 수정(error correction), read-ahead 등의 내부 상태를 조회하는 기능
- SSD 내부 동작의 제어 기능
  - SSD 내부 동작을 제어할 수 있다면, 응용 상황에 맞게 SSD 를 사용할 수 있음
  - Ex) GC 수행 기준이나 주기를 제어하여, latency 또는 throughput 성능을 조절하거나 GC 의 일시적 중지/재개를 통해 쓰기 요청과 GC 작업의 간섭 최소화
- Transparent compression
  - 파일 저장 목적으로 사용하려면 SSD가 자동압축/해제 기능을 지원
- Over-provisioning
  - SSD 용량을 full로 사용하더라도 성능 저하나 고장이 발생하지 않도록 SSD 내부의 예비공간을 충분히 제공

박준현, 신기빈, 송창현, “IDC에서의 애플리케이션 이슈와 SSD 요구 특성”,
전자공학회지(The Magazine of the IEEK) 제41권 5호, pp.57-67, 2014년 5월