ARCUS CACHE CLOUD

캐시 개요, 기술 특징, 활용 방안

Remote memory caching
used by service applications
for look-aside cache
2020.01

JaM2in

순서

JaM2in

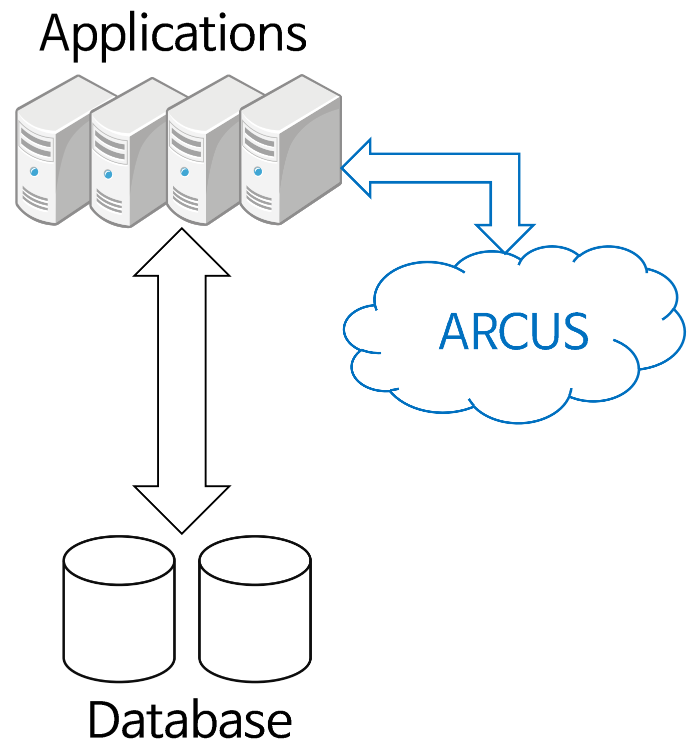
ARCUS 개요

아커스는 NAVER 서비스의 요구 사항을 반영하여 개발한 메모리 캐시 클러스터 시스템으로 응용 수준에서의 데이터 캐싱을 구현합니다.

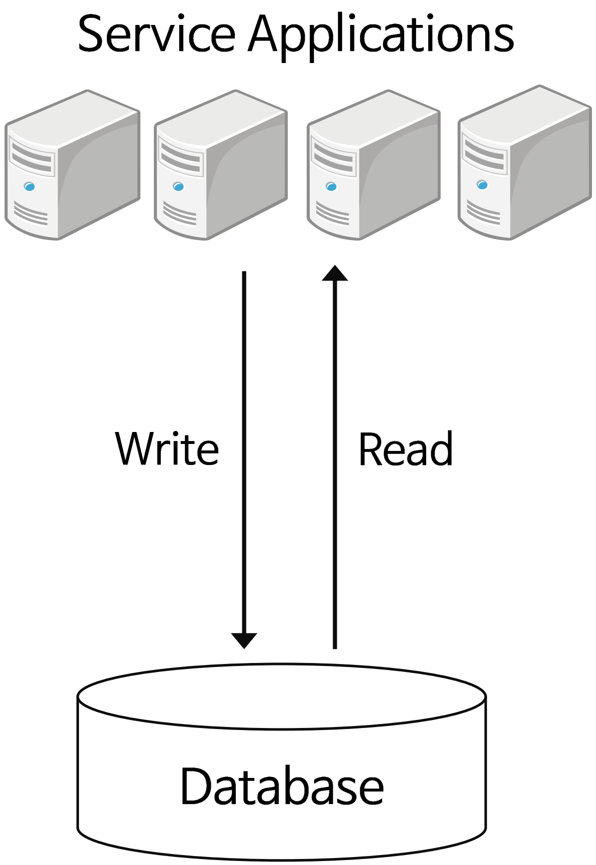
- (조회) 요청 처리 성능을 극대화
  - 고성능 제공

    높은 처리용량: 초당 100K / 1 노드

    빠른 응답속도: 평균 1 millisecond
  - 수평 확장이 가능 (운영 도중)
- 3,000+ instances, 200+ clusters



캐시 필요성 : 기존 구조의 이슈



Large-scale 웹 서비스

- 사용자 증가 (요청량 증가)
- 데이터 증가

성능 이슈

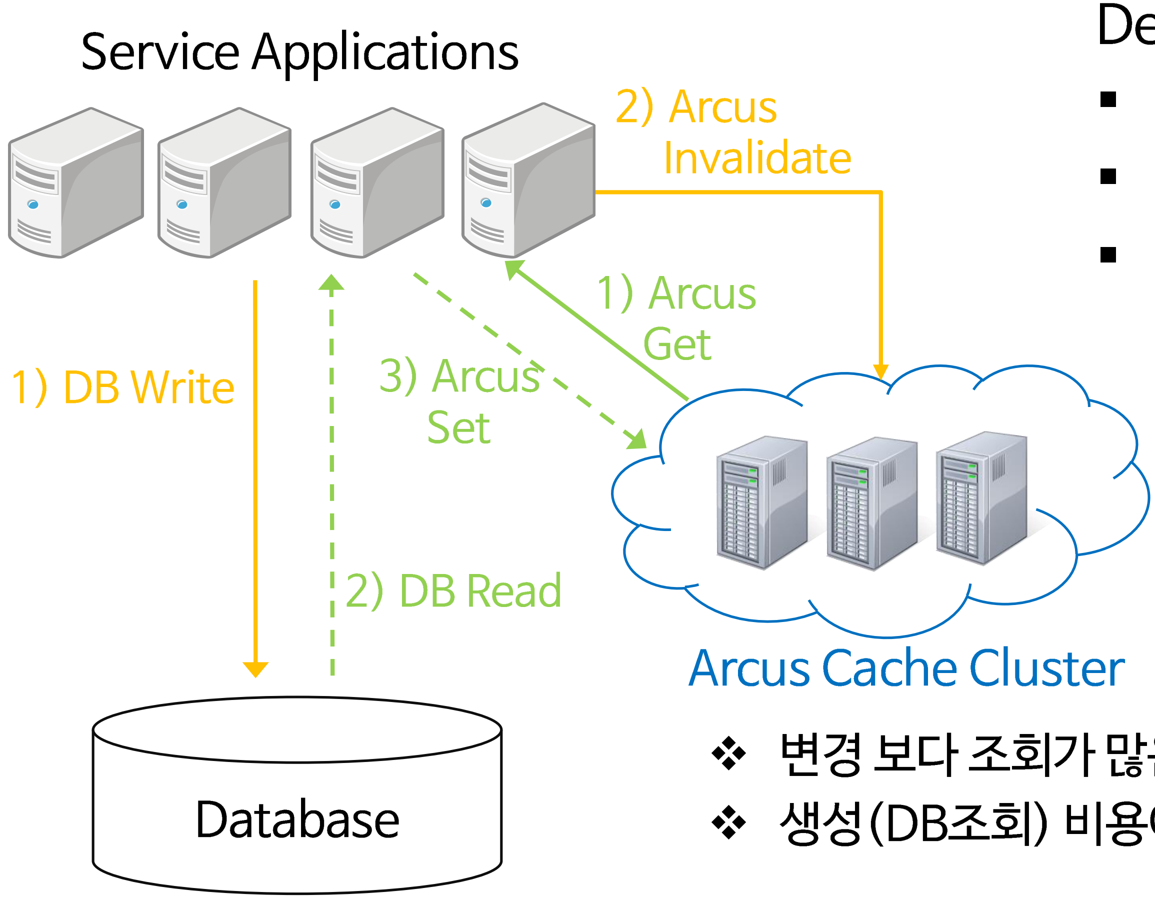
- 느린 응답, 처리량 부족
- DB scale-out 어려움
- 급변하는 요청 처리 미흡(예, 이벤트)

고비용 이슈

- DB 라이선스, 장비

JaM2in

캐시 필요성: DB + ARCUS 캐시



Demand-Fill 방식

- DB 조회 결과
- 서비스 API 수행 결과
- …

❖ 변경 보다 조회가 많은 데이터
❖ 생성(DB조회) 비용이 큰 데이터

ARCUS 캐시의 효과

DB Only 구성 대비 DB + ARCUS 캐시 구성의 효과

1) 응답속도 개선
   - 0.01초 이하의 빠른 응답으로, 쾌적한 서비스 구현
2) 요청처리량 확대
   - 클러스터 크기 만큼의 요청 처리량: (100K * 캐시노드수)/1초
3) 실시간 급증 요청 처리 (이벤트, 사회적 이슈에 의한)
   - 운영 도중에 캐시 클러스터 확장/축소 가능
4) 저비용 및 안정화 구현
   - DB 부하 감소/안정화로 DB 관련 비용(인력 포함) 절감
   - 처리량 대비 적은 비용

JaM2in

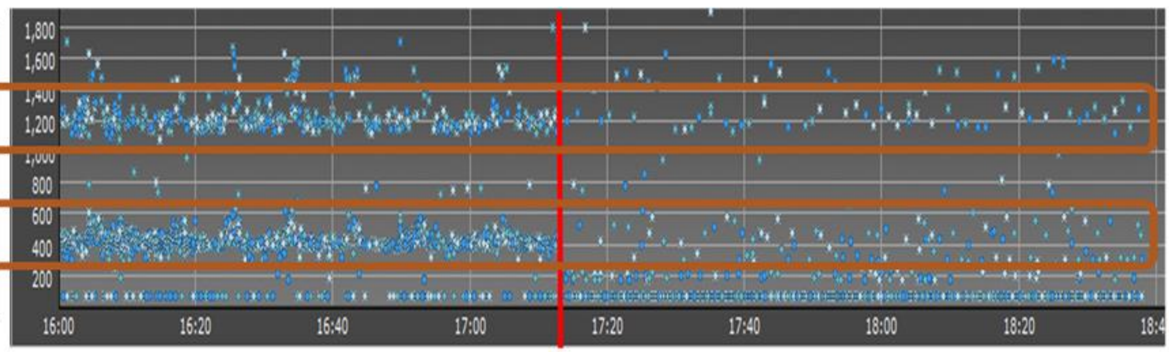
ARCUS 응답속도 개선 사례

고객사의 특정 서비스 API에 대해 Spring AOP 방식으로 ARCUS 캐시 적용

- 서비스 응답속도: 76.1% 개선

구분	개선 전	개선 후	차이
1.2초 이상	96	18	-78
1.0초 이하	4	4	0
0.5초 이하	272	17	-255
0.1초 이하	37	385	348

개선 전후 응답 시간별 건수 비교 (20분간)

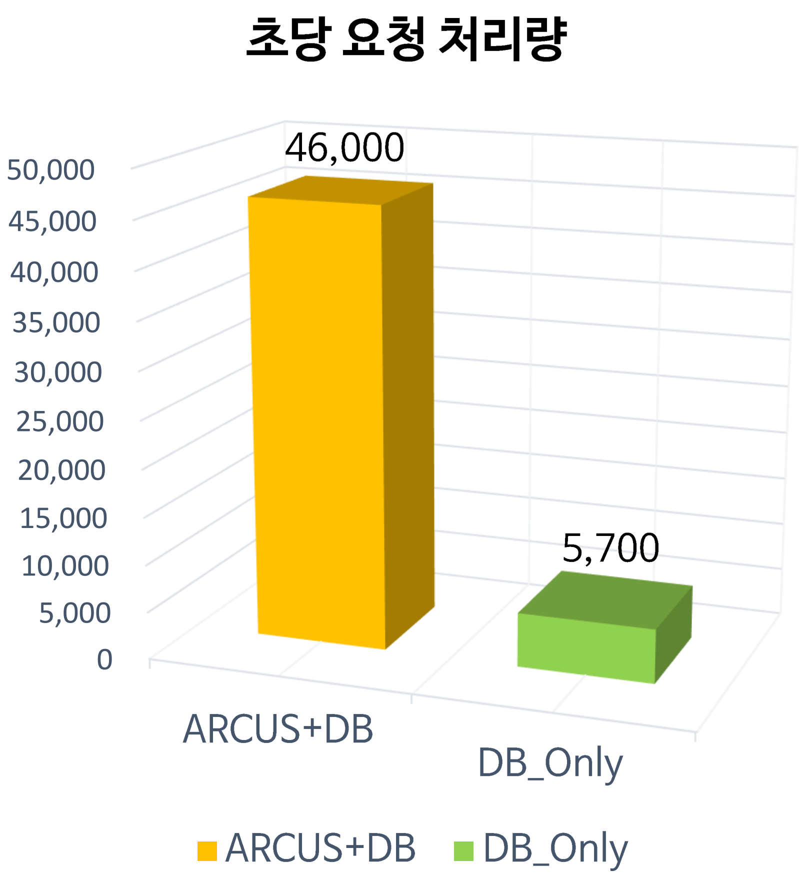


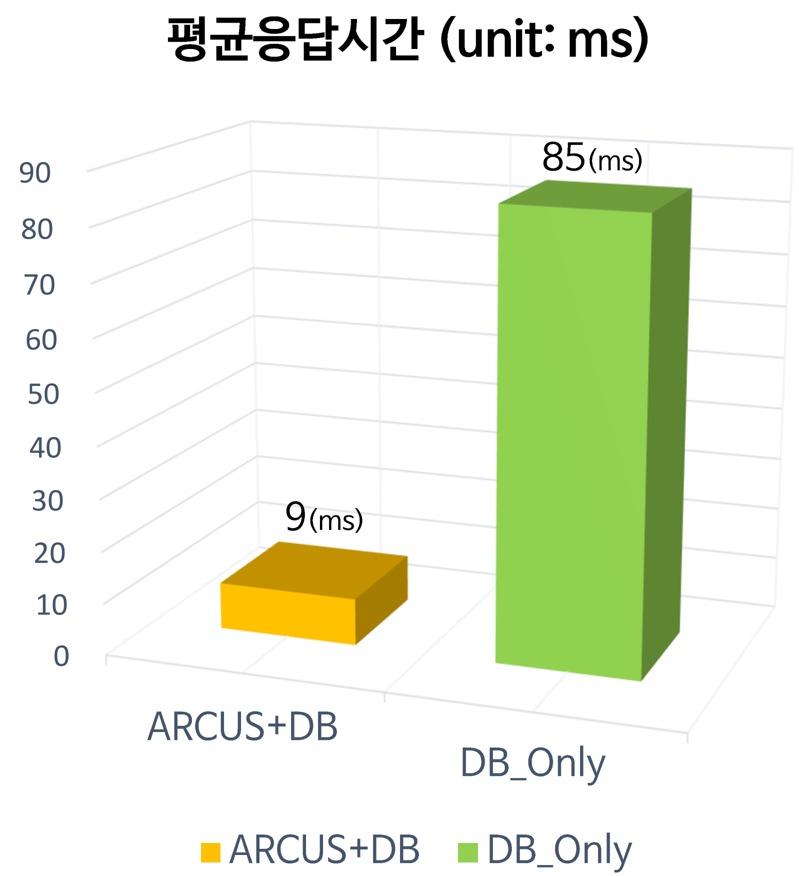
ARCUS 캐시 적용 전 ARCUS 캐시 적용 후

서비스 응답속도 개선은
서비스 이용에 쾌적함을 제공하여 보다 많은 정보가 이용자에게 노출되게 하며,
이는 서비스 만족도 향상 및 매출 증대로 이어지게 합니다.

ARCUS 성능

- [가상 시나리오] 상품의 등록과 조회, 판매자 상품 리스트 조회 등





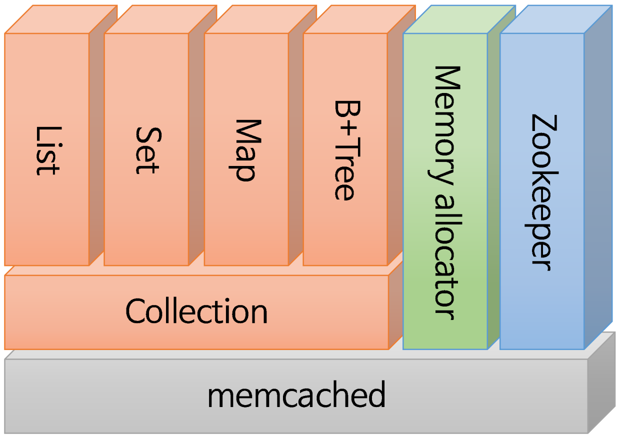
JaM2in

순서

1) ARCUS 캐시 개요
2) ARCUS 기술 특징
3) ARCUS 적용 방안
4) ARCUS 기타

ARCUS 주요 기술 특징

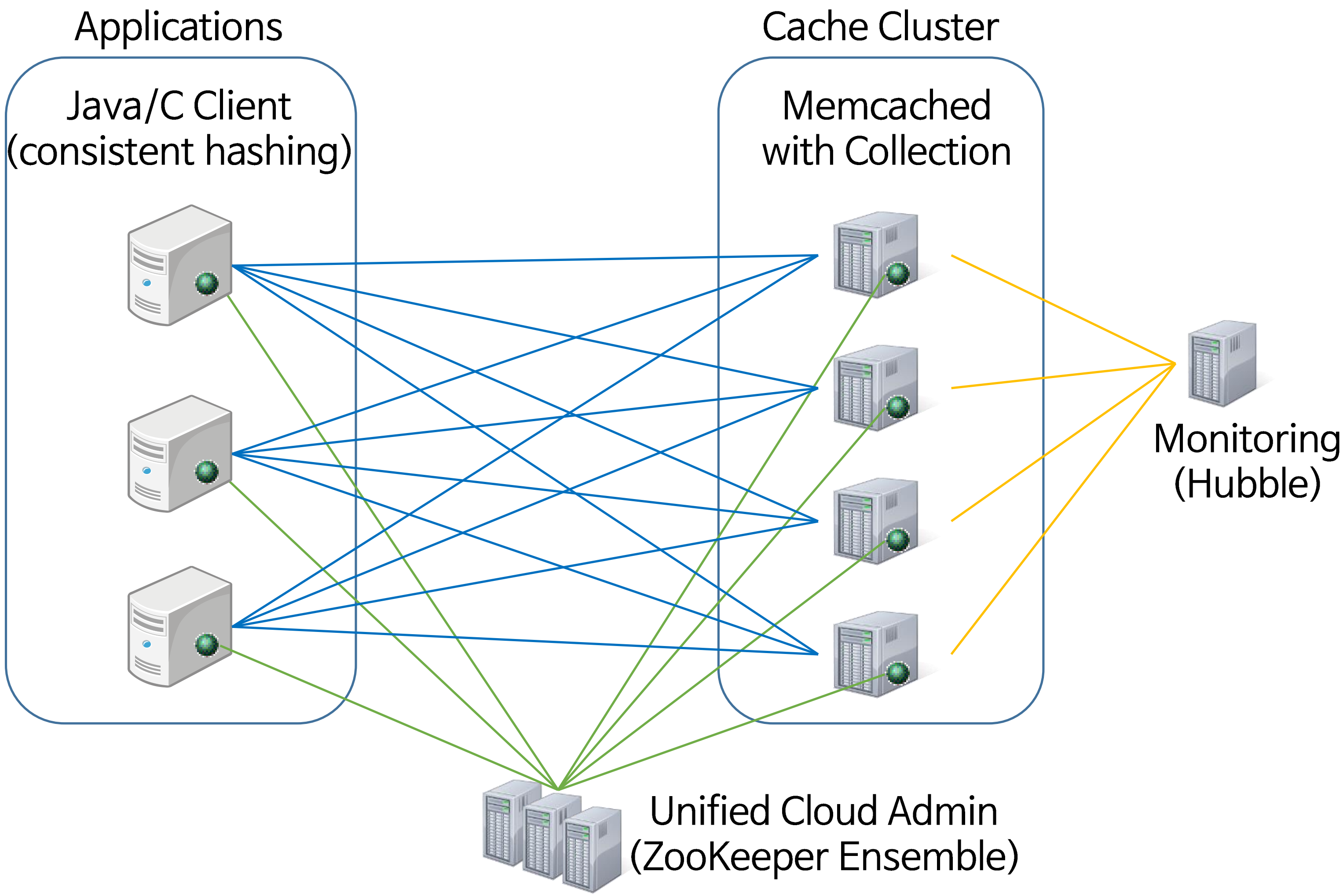
- 메모리 캐시로 고성능 제공 (memcached 기반)
  - 100~200K requests/sec (1 cache node)
  - 평균 1ms 이하 응답속도
- 확장된 Key-Value 데이터 모델
  - Collection(list/set/map/b+tree) 데이터 지원
  - B+Tree 이용한 차별화 기능 제공
- Elastic Cache Cluster 구현 (ZooKeeper 기반)
  - Scale-out, Automatic fail-stop
- 그 외의 주요 특징들
  - TTL(time-to-live), LRU 기반 eviction
  - Cache 용도에 최적화된 메모리 관리
  - Cache 데이터 이중화 : Semi-Sync 복제
  - Cache 장비와 프로세스의 모니터링 기능
  - 쿠버네티스 환경에서의 배포 및 운영 지원



Arcus-memcached 노드

JaM2in

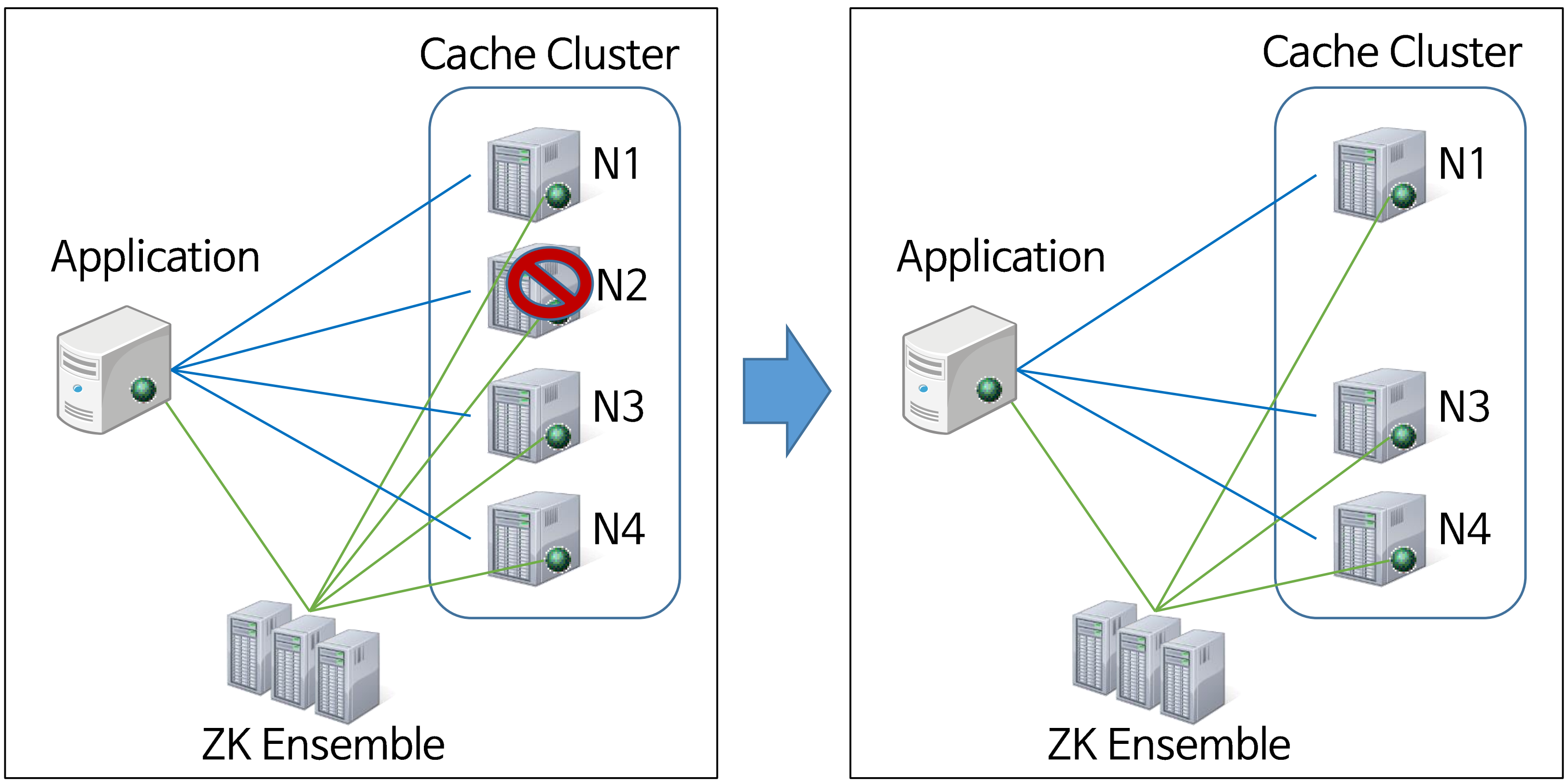
ARCUS Architecture



JaM2in

Automatic Fail-Stop

- 캐시 노드의 장애 탐지 및 자동 제거 : down, hang, 네트워크 단절
- 캐시 데이터의 자동 재배치가 진행되어 캐시 서비스의 정상화 유지



JaM2in

ARCUS 메모리 관리

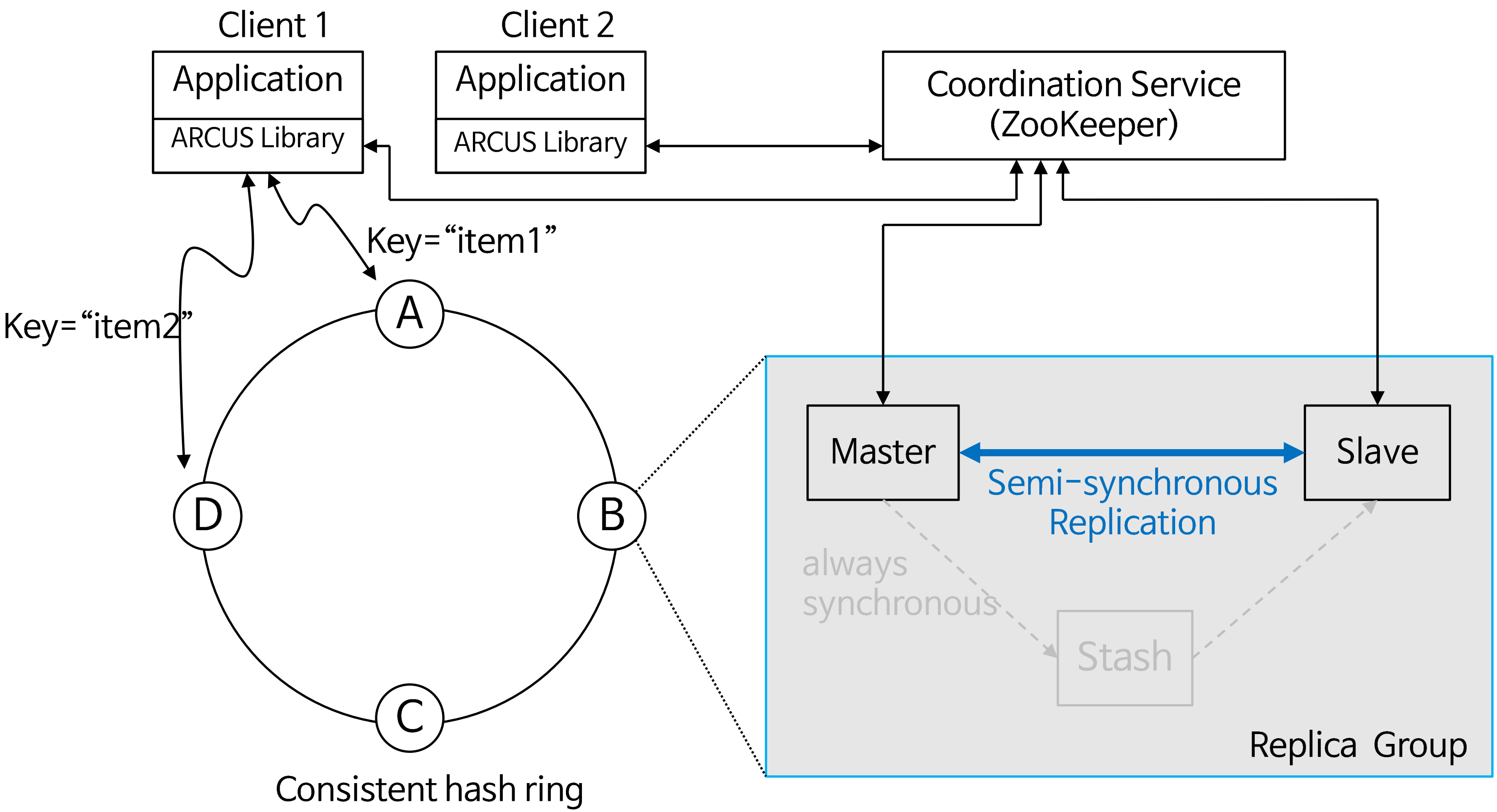
- ARCUS : cache 용도에 최적화된 메모리 관리자를 자체 구현
  - ➢ 메모리 파편화 최소화

- REDIS => ARCUS 교체 사례 : Redis 메모리 파편화 (2014 DEVIEW)
  - ➢ 실제 응용 데이터(2.6GB) << Redis 실사용 메모리(12GB)
  - ➢ Swap 영역 사용 => 아주 느린 응답 발생 : 재구동 조치

M	6,596,618	50,000	565	2.29M	4.80M	81,985	8.00G	2.66G	12.	2.8.6
M)	6,191,293	50,000	558	2.22M	5.84M	79,424	8.00G	2.55G	12.	2.8.6
M	6,109,388	50,000	564	2.23M	5.10M	79,260	8.00G	2.56G	12.	2.8.6
M)	5,683,407	50,000	545	2.09M	4.78M	73,412	8.00G	2.44G	12.	2.8.6

캐시 데이터 이중화 (ARCUS Enterprise)

- Failover 시에 데이터 손실(data loss)이 없는 복제 기능



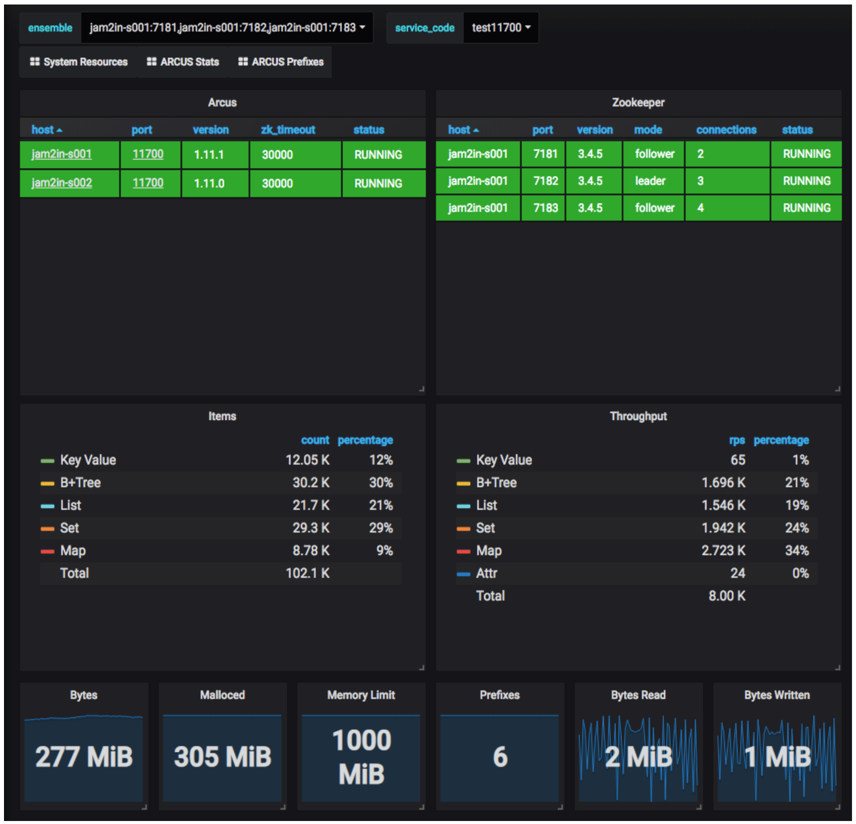
JaM2in

Replication Features (ARCUS Enterprise)

- Automatic Failover
  - Master 장애 시, Slave가 자동으로 Master로 승격
- Switchover
  - Master와 Slave의 role 교체
  - 버전 업그레이드, 장비 교체를 위한 운영에서 필요
- Network Throttling
  - 1개 NIC으로 서비스와 복제를 함께 수행하는 환경
  - 복제의 network bandwidth 제한 (예, 50MB/sec in 1Gbps)
- Client의 read replica 선택 기능
  - Client에서 read replica (master or slave) 선택 기능

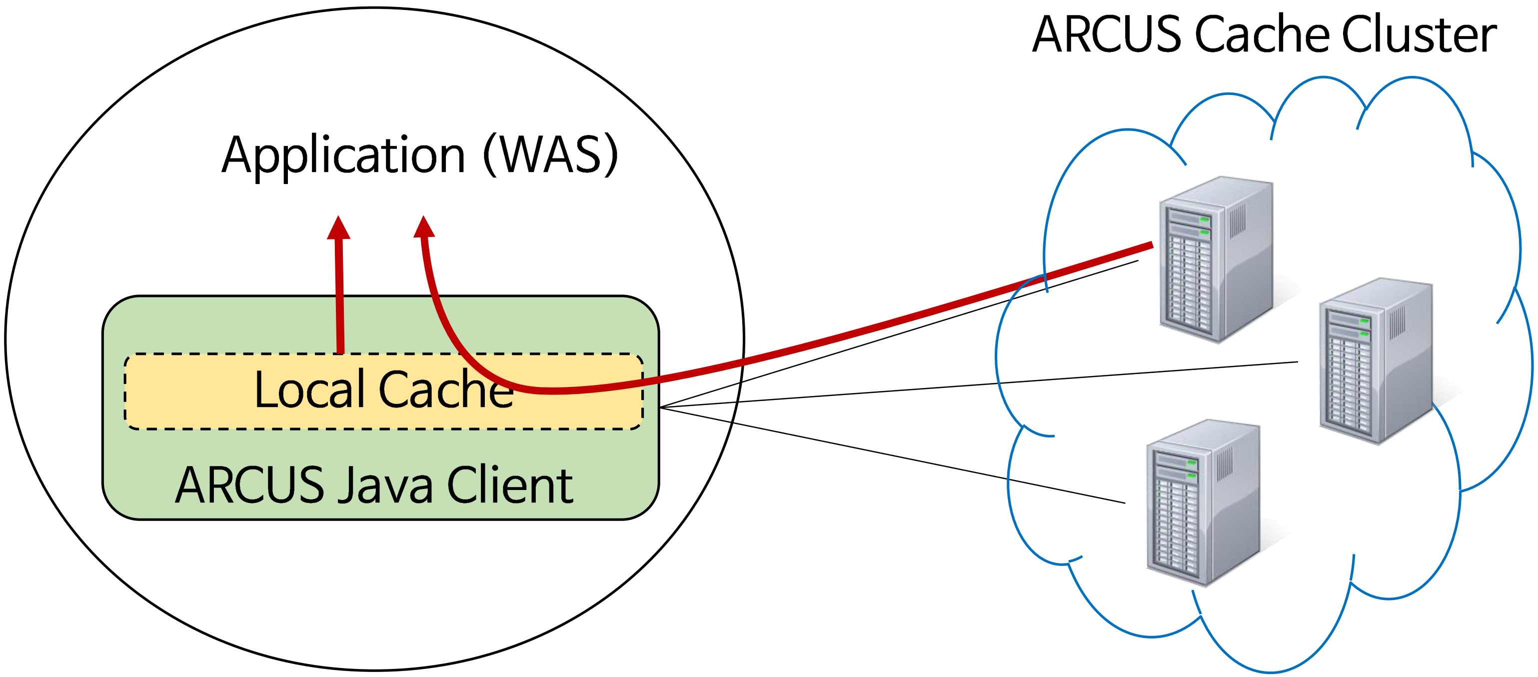
ARCUS Hubble (ARCUS Enterprise)

- Cache 장비 & 프로세스의 사용량 상태를 모니터링



Local Caching (at ARCUS Java Client)

- 데이터 압축
- Local Cache (or Front Cache) : disabled (default)
  - 적은 수의 hot data 캐싱 용도



JaM2in

순서

1) ARCUS 캐시 개요
2) ARCUS 기술 특징
3) ARCUS 적용 방안
4) ARCUS 기타

조회 성능 극대화

- 카테고리 별 주요 상품 목록
- 특정성향 별 추천 상품 목록
- 상품 상세 정보
- 주요 뉴스 내용
- 그 외, 여러 사용자가 빈번히 조회하는 데이터

- 가장 보편적으로 많이 활용
  - 대부분 KV 유형
  - 일부는 Collection 유형

Counting (KV)

- 상품의 좋아요 개수
- PV, UV 계산
- 로그인 실패 횟수
- 특정 IP에서 과도한 요청 확인
- API 호출 제한

user:100

로그인 실패 →

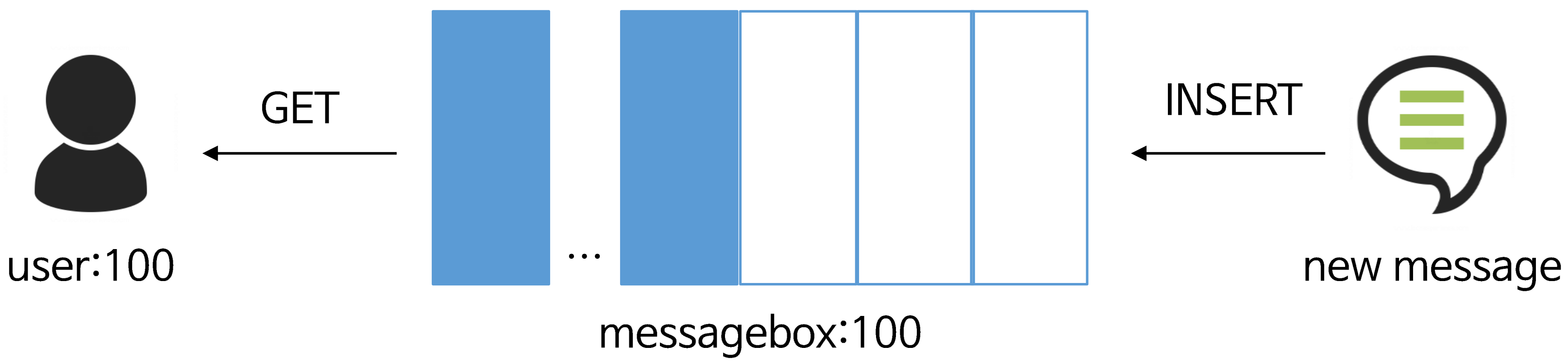
KEY	VALUE
loginfail:100	3

INCR →

KEY	VALUE
loginfail:100	4

Message Box (List)

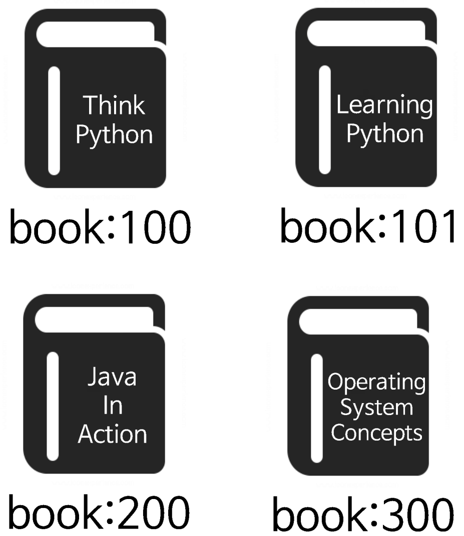
- 메시지 서비스에서 최근 도착한 N개의 메시지를 저장
- 유저가 메시지 박스로부터 최근 메시지를 읽으면, 박스 안의 메시지를 삭제



JaM2in

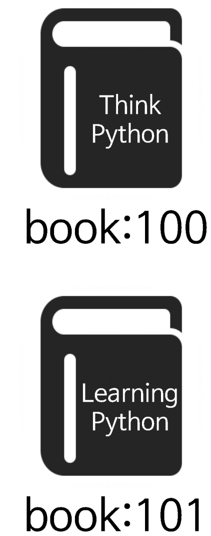
태그 분류 (Set)

- 특정 Tag와 관련된 상품코드들을 Set에 삽입
- Tag의 상품 코드를 가져옴으로써 Tag 검색에 활용



KEY	VALUES
tag:python	100, 200
tag:java	200
tag:language	100, 101, 200
tag:os	300

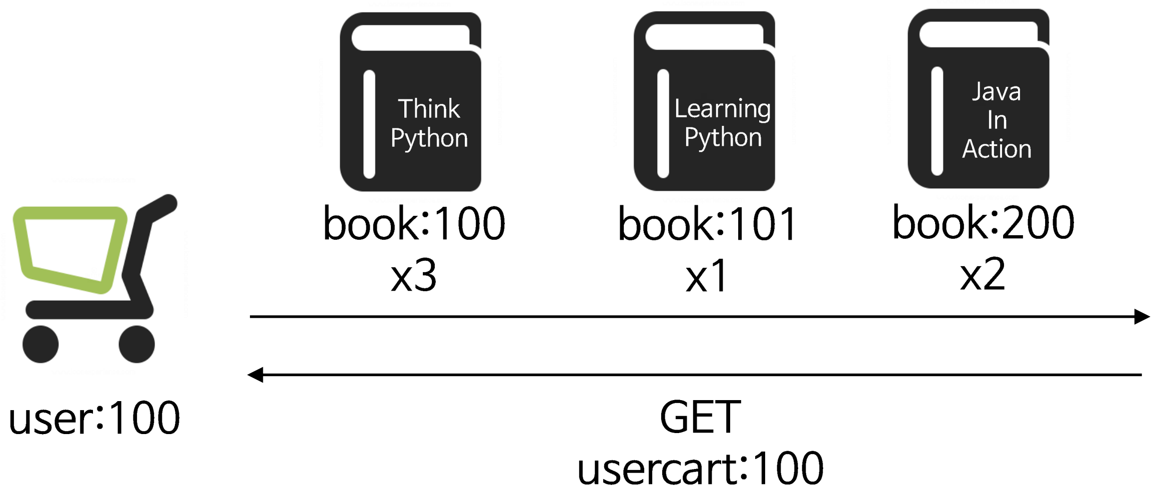
GET tag:python



JaM2in

장바구니 관리 (Map)

- 유저 별로 장바구니에 담긴 상품들을 Map으로 관리
- 메모리 낭비를 줄이기 위해 장바구니 만료 시간(TTL) 설정이 요구됨



상품 ID (KEY)	수량 (VALUE)
100	3
101	1
200	2

Leaderboard, Rank (B+Tree)

- 항상 정렬된 데이터로 관리 가능
  - 판매량 순으로 상품 조회
  - 스코어 순으로 게임 랭킹 조회
- 특정 순위의 범위 검색(예: 10~30위), 조건 검색

GET
bestsellers:201901

판매수 (BKEY)	상품 ID (DATA)
5000	100
4000	101
3000	200

JaM2in

Game Ranking (B+Tree)

- Top 10 ‹점수, 유저› 조회
- 나의 순위 조회
- 나의 점수 앞뒤에서 각 5개의 ‹점수, 유저› 조회

순위	점수 (BKEY)	UserID (DATA)
1	5000	100
2	4980	101
	…	…
98	4500	666
99	4470	444
100	4400	555
101	4390	222
102	4380	333
	…	…
N	1000	102

JaM2in

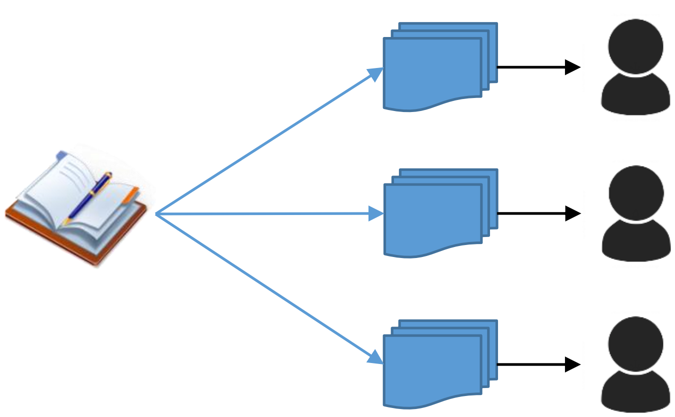
SNS 모아보기 (B+Tree)

- "게시글 모아보기" DB 질의를 수행할 경우 (feed list 크면, 성능 저하 초래)

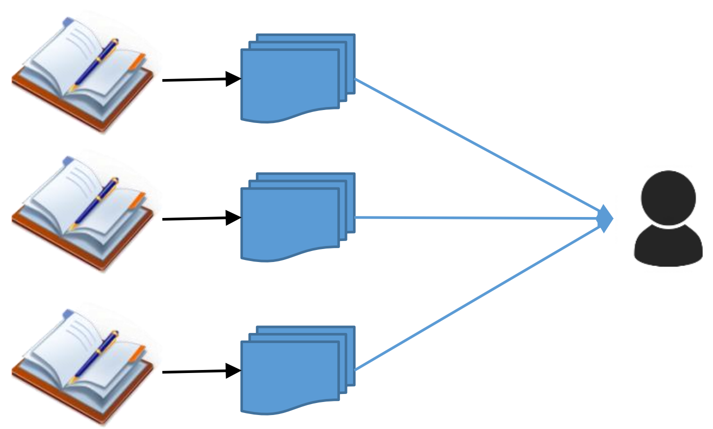
```
SELECT * FROM posts
WHERE feedid IN (feedID1, feedID2, …, feedIDn) AND create_time 〈 sysdate()
ORDER BY create_time DESC
LIMIT 20;
```

- ARCUS 캐시 적용

Push - 유저별 Inbox Cache 구현



Pull – 피드별 Outbox Cache 구현



순서

1) ARCUS 캐시 개요
2) ARCUS 기술 특징
3) ARCUS 활용 방안
4) ARCUS 기타
    A. 캐시 공통 모듈
    B. 개발 로드맵
    C. Redis 와의 비교
    D. 잼투인 소개

ARCUS 캐시 적용 – 캐시 공통 모듈

- 특정 서비스 API (or method) 호출 시, ARCUS 캐시 적용

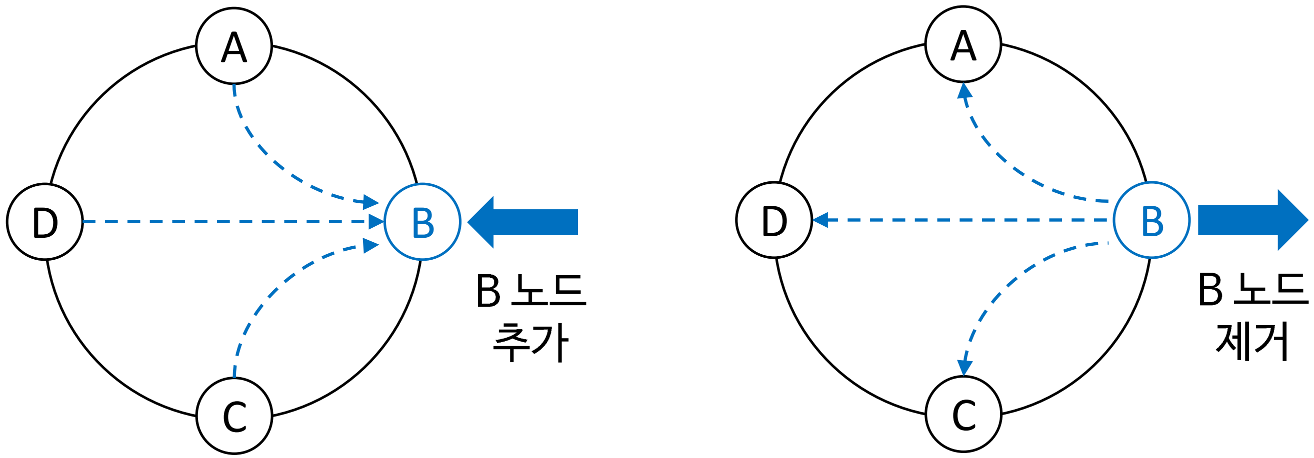
```
serviceAPI(arguments)
{
  result = Cache_Get(key="API(or method) name : argument values");
  if (result == null) { /* cache miss */
     result = Execute the original logic including DB Read;
     Cache_Set(key, result);
  }
  return result;
}
```

- 응용 수정 없이 위의 캐싱 로직을 자동 생성하는 ARCUS 캐시 공통 모듈 제공
  - ➢ Spring AOP – Property 방식과 Annotation 방식
  - ➢ 응용 재배포 없이 캐시 대상 서비스 API를 변경하는 방안도 제공함.

JaM2in

Data Migration 기능 – 2020 릴리즈 예정

- 캐시 클러스터의 scale in/out 시, 데이터 이관 기능



- 기대 효과
  - 캐시 미스 방지 => HA 서비스 제공
  - Stale(or old) 캐시 데이터의 제거

JaM2in

Persistence 기능 - 2020 릴리즈 예정

- Snapshot과 Command Logging 기반의 데이터 저장과 복구

Snapshot

- 전체 데이터의 백업 본사본을 생성하는 작업
- 주기적으로 수행하여 명령 로그 크기를 줄임

Command Logging

- 데이터 변경의 로깅으로 변경 요청 자체를 기록
- 동기(sync) 로깅과 비동기(async) 로깅 지원
- 동기 로깅에서도 고성능 제공

데이터 복구

Snapshot 데이터와 Command Log의 재수행(replay) 방식으로 데이터 복구

JaM2in

ARCUS vs. REDIS

항목	ARCUS	REDIS
분산 캐시 인지도	국내에서 일부	매우 높음
고성능과 수평 확장	O	O
데이터 콜렉션 지원	List/Set/Map/B+Tree	List/Set/Hash/SortedSet Geohash, Stream, …
메모리 파편화 이슈	적음 (캐시 전용 메모리 관리자)	많음 (일반 메모리 관리자)
내고장성	O (automatic failstop)	X (고장 노드 제거 필요)
스레드 구조	다중 스레드 구조	단일 스레드 구조
복제 방식	1:1, Semi-Sync	1:N, Async
Scale in/out 시의 데이터 이관	X (PoC 완료, Automatic 방식)	O (Semi-Automatic)
제품 개발 배경 차이	운영 안정성 추구 (long request 제약 있음)	개발 편리성 추구 (정확한 내부 동작 이해 필요)
국내 기술지원 능력	매우 높음 (개발자 직접 지원, 다년간 경험)	낮음

JaM2in

잼투인㈜ 소개

JaM2in

잼투인㈜은 아카스(ARCUS)개발 핵심 인력이 창업한 회사로, ARCUS 개발/컨설팅/기술지원을 전문적으로 제공하면서 다양한 NoSQL 개발하는 **기술 중심 회사**입니다.

2014~5

잼투인㈜ 설립(2014.07)
잼투인-큐브리드 협력사 체결
네이버, 라인, 밴드, 카카오스토리 등 ARCUS 사용 확대

2016~7

기업부설연구소 설립
벤처기업 인증
롯데홈쇼핑 ARCUS 구축

2018~9

파스-타(PaaS-TA) 공동연구협력
하이마트 ARCUS 구축
올리브영 ARCUS 구축

감사합니다.

잼투인㈜ www.jam2in.com

ARCUS 문의: support@jam2in.com