

Arcus Collection 기능과 Open Source 전략

박준현 / Naver Labs

2014. 03. 13

N A V E R | L | A | B | S |

목차

1. Arcus 소개
2. Arcus Collection (특히, b+tree collection)
3. Arcus Open Source

1. Arcus 소개

Arcus ?

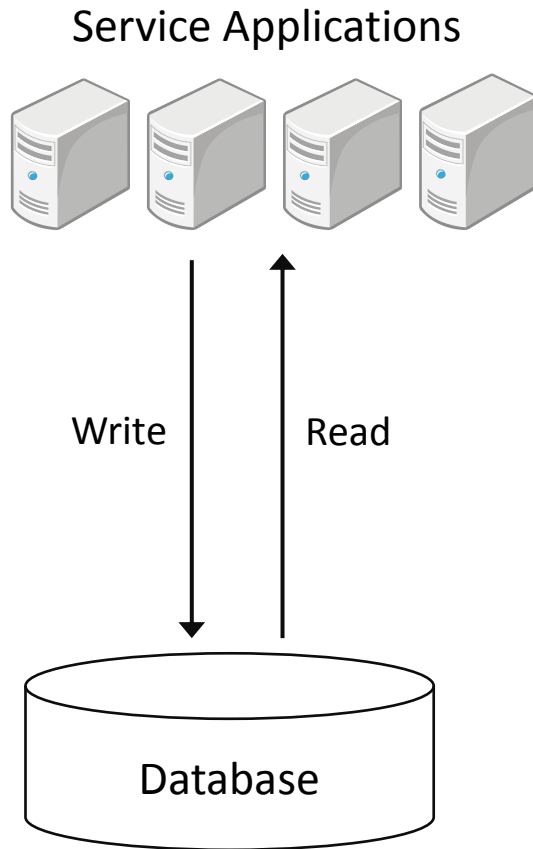
- Memory Cache Cloud (developed by Naver Corp.)
 - ✓ memcached 기반의 key-value cache cloud
 - ✓ In-house development : 2009 ~ 현재

- Arcus 이름
 - ✓ 아치 구름

A R C U S
c a c h e c l o u d

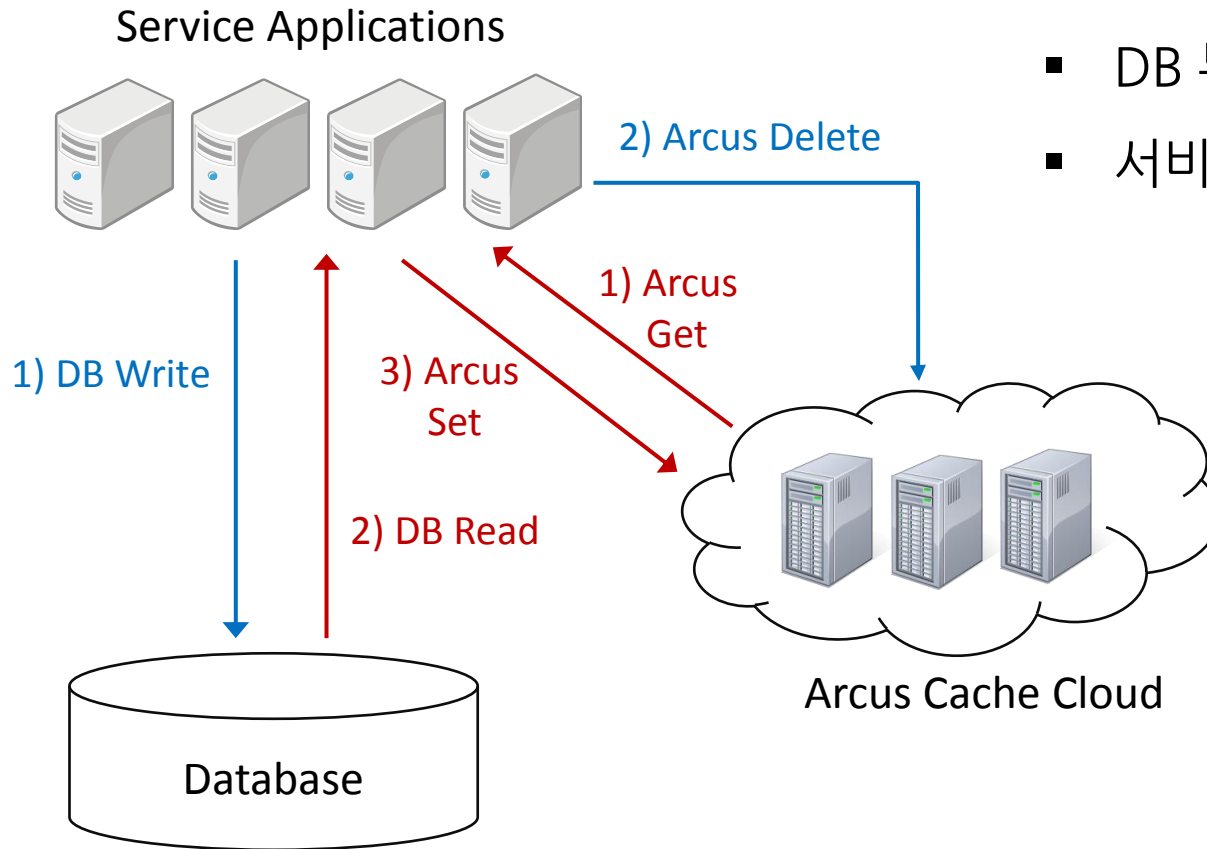


Arcus 개발 배경



- 데이터 양과 요청 양의 증가
- 성능 이슈
 - ✓ Throughput
 - ✓ Response Time
- Scalability 이슈

Arcus 개발 배경 - Arcus 사용 구조

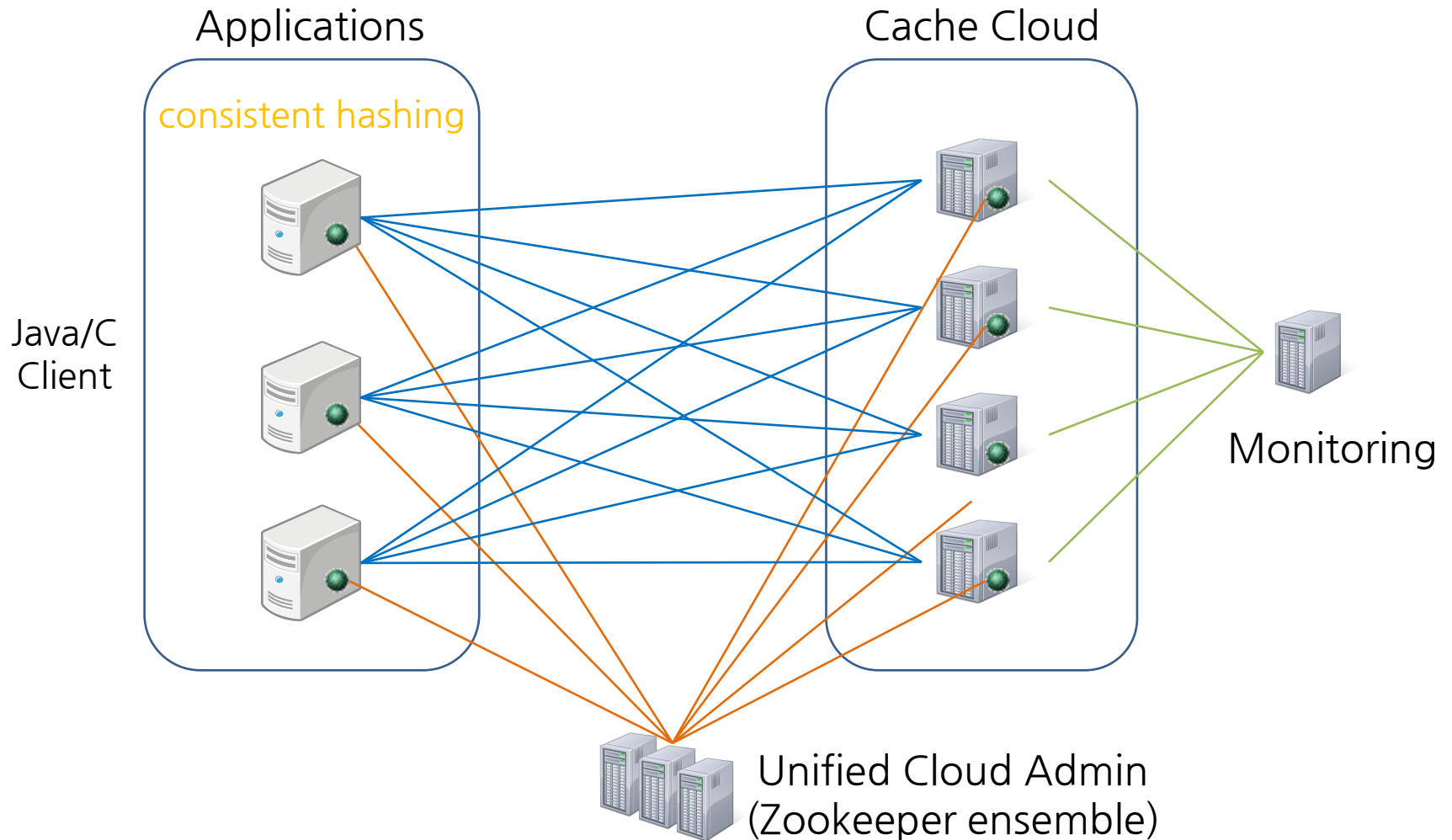


- 성능 해결
- DB 부하 경감
- 서비스 요구 반영

Memcached 대비 Arcus 개발 사항들

- Zookeeper 기반의 Cloud 형성 및 관리
 - ✓ Elastic - node 추가/삭제, automatic failover
- Prefix 기능 확장 - delete, stats
 - ✓ A prefix in key string : <key> => <prefix>:<subkey>
- Collection - List / Set / B+tree
- Small memory allocator
- Sticky Item
- Eager Item Expiration
- 동적 configuration(memlimit, maxconns, ...) 변경
- Item 속성 조회/변경
- Front caching at client, ...

Arcus Architecture



Arcus Stats

- The number of clouds
 - ✓ 대략 140 개
- The total memory used
 - ✓ 대략 10 TB
- The number of operations
 - ✓ 220만 ops / sec (오후 3시경)
 - ✓ $220 * 3600 * 16 \approx 1267$ 억 ops / day

Arcus Misc

- Supported OS - Linux Only
 - ✓ CentOS 64 bit
- Supported Clients
 - ✓ Java, C
- Supported Server Protocol
 - ✓ Ascii protocol only

2. Arcus Collection (B+Tree Collection)

Arcus Collection

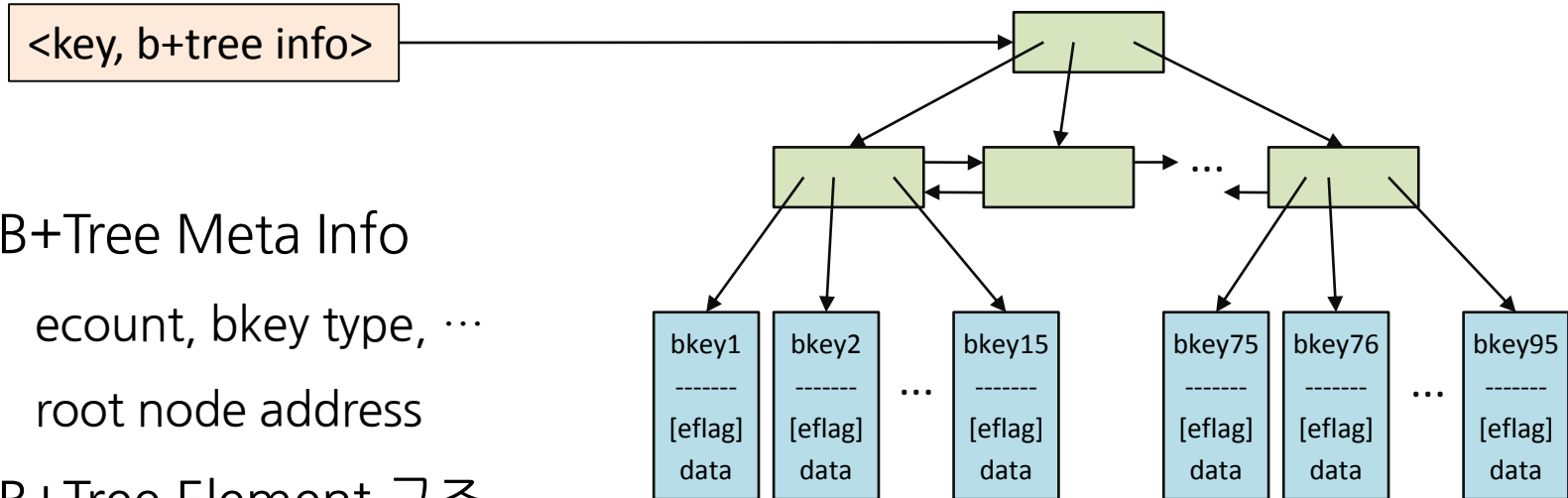
- Simple Key-Value Item
 - ✓ Item : < key, **single data** >
 - ✓ Operation : get/set/delete/...
- Collection item
 - ✓ Item: < key, **a collection of elements** >
 - ✓ Element : < single data >
 - ✓ Operation : create/drop/insert/delete/get/...

Collection 유형

유형	특징
List	▪ Double linked list 구조 ▪ List index 기반의 element 접근
Set	▪ An unordered set of unique data ▪ Membership checking (예, 친구 정보) ▪ Extendable hash table 구조
B+Tree	▪ An ordered data set based on b+tree key ▪ bkey(b+tree key) 기반의 range scan (with filtering) ▪ Position 기반의 range scan

B+Tree Collection 구조

■ B+Tree Collection 구조



■ B+Tree Meta Info

- ✓ ecount, bkey type, ...
- ✓ root node address

■ B+Tree Element 구조

bkey	■ B+tree key, unique value in b+tree. ■ 8 bytes unsigned integer / variable length(1~31) bytes array
[eflag]	■ Optional element flag, used as filterable field ■ Variable length(1~31) bytes array
data	■ 단순 저장/조회 용도의 data field (max 4KB)

B+Tree Collection 속성

- B+tree collection 속성
 - ✓ maxcount, maxbkeyrange, overflowaction, ...
- maxbkeyrange
 - ✓ Maximum bkey range allowed, default: 0(disabled)
 - ✓ 예) 시간 역순의 post 저장 b+tree에 최근 2일치 데이터만 보관
 - maxbkeyrange를 2일치로 설정 (bkey는 post 생성 시간 가정)
 - New bkey 추가 => maxbkeyrange 검사 및 old bkey 자동 삭제
- overflowaction
 - ✓ smallest_trim, largest_trim, error

Arcus Collection 연산

- B+tree collection 연산

- ✓ create / drop

- B+tree element 연산

- ✓ insert/upsert

- ✓ update

- ✓ delete

- ✓ get

- ✓ count

- ✓ incr/decr

- ✓ find position

- ✓ get by position

Bkey 기반의 조회

- B+tree get
 - ✓ <key, bkey_range, [eflag_filter,] [[offset,] count]> => elements
- 조회 조건

조건	설명
bkey_range	▪ Mandatory, ascending or descending 조회 가능 ▪ Ex) 100..200, 200..100, 0x00AA..0x00FF
[eflag_filter]	▪ Optional filter condition ▪ eflag 부분 값/전체 값에 대한 filter 조건 ▪ [bitwise operator +] comparison operator ✓ bitwise : AND, OR, XOR ✓ comparison: EQ, NE, LT, LE, GT, GE, IN, NOT IN
[[offset,count]	▪ Optional, skip count 와 retrieval count 명시

Position 기반의 연산

- B+tree find position
 - ✓ $\langle \text{key}, \text{bkey}, \text{order} \rangle \Rightarrow \text{position}$
 - ✓ $\langle \text{order} \rangle$: asc | desc
- B+tree get by position
 - ✓ $\langle \text{key}, \text{order}, \text{position_range} \rangle \Rightarrow \text{elements}$
 - ✓ $\langle \text{order} \rangle$: asc | desc

참고) Position 기반 연산의 제공을 위해, b+tree nonleaf node는 각 lower sub-tree에 존재하는 # of elements 정보를 유지

B+Tree 활용 1) Bkey 기반의 조회

- SNS의 timeline 서비스들
- B+tree element 정보
 - ✓ Bkey: post 생성 시간
 - ✓ Eflag: post category 표시
 - ✓ Data: post 정보
- 조회 요청 예들
 - ✓ Example 1) 최근 20개 post 조회
 - Get : <key, 20201231..19000101, 20>
 - ✓ Example 2) 음악, 미술에 관한 최근 20개 post 조회
 - Get : <key, 20201231..19000101, 0, EQ, “음악, 미술”, 20>

B+Tree 활용 2) Position 기반의 조회

- Ranking 시스템
- B+tree element 정보
 - ✓ Bkey: score
 - ✓ Data: User info
- 조회 요청 예들
 - ✓ Example 1) Score 순위 기준 101등 ~ 120등의 사용자 조회
 - Get by position: <key, desc, 101..120>
 - ✓ Example 2) 특정 score 앞에 10명의 사용자 조회
 - Find position: <key, bkey, desc> => 300 이라 가정
 - Get by position <key, desc, 290..299>

B+Tree 활용 3) Record 구조

- 하나의 Record 구조 구현
 - ✓ $\langle \text{field name}, \text{value} \rangle, \dots \langle \text{field name}, \text{value} \rangle$
 - ✓ Field name: bkey로 encoding
 - ✓ Field value: data 부분에 저장
- Record 조회
 - ✓ $\langle \text{key} \rangle$ 로 특정 record 선택
 - ✓ 전체 field 조회 : bkey range를 최소, 최대 값으로 지정
 - ✓ 일부 field 조회 : single bkey로 조회 또는 bkey range로 조회
 - 함께 조회할 field name은 bkey comparison 상 서로 이웃하도록 encoding 필요

3. Arcus Open Source

네이버의 Open Source 참여

- Arcus Open Source 준비 중
 - ✓ 네이버도 많은 Open Source SW를 사용하여 서비스를 개발
 - ✓ 따라서, Open Source SW 생산자로서의 기여는 당연한 의무

- 네이버의 Open Source SW들
 - ✓ Cubrid - Disk based DBMS
 - ✓ XpressEngine - Contents Management System
 - ✓ nGrinder - 성능 측정 툴
 - ✓ Jindo - Javascript Library
 - ✓ 그 외에도 여러 OSS들

Arcus Open Source

- Open 예정 : 1~2달 내
- Open 내용
 - ✓ Arcus cache server & Java/C clients
 - ✓ Zookeeper library with Arcus modification
 - ✓ Arcus monitoring system, ...
- 기대 효과
 - ✓ 서비스 개발에서 DBMS 외에 Cache Solution은 필수
 - ✓ Arcus는 국내 최초의 Open Source Cache Solution.
 - ✓ 국내 여러 서비스 개발 및 품질 향상에 기여할 것으로 기대.
 - ✓ Open Source SW 발전 및 개발자 양성

Arcus 개발자로서의 바람

- 많은 서비스들에서 확산 적용
 - ✓ 사용되지 않는 Arcus는 의미가 없음
- Open Source SW 자체의 기능, 품질, 사용편이성
- Open Source SW 인식의 중요성
 - ✓ Not Free SW => 공유와 참여로 발전
 - ✓ Maintainer, Contribution, Community, ... 활성화될 수 있게.
- 지속적인 발전
 - ✓ (Open Source) System SW와 서비스 개발의 수평적 도움 관계
 - ✓ 안정되고 자유로운 SW 개발 환경 조성 및 SW 개발자 양성
 - ✓ SW 기술 및 품질 향상 => 여러 서비스 개발에 기여

Thank You !!