ARCUS(Memory Cache Cloud System) Overview and Use Cases

Taeho Ahn
(thahn999@jam2in.com)

2015. 11. 17

JaM2in

Part 1. ARCUS Overview

JaM2in – 잼투인㈜

- Founded by ARCUS core developer last year.
  - ✓ Facebook Page – https://www.facebook.com/jam2in
- Main Business
  - ✓ Development: ARCUS, NoSQL, …
  - ✓ Consulting, technical support, …
- Future Dev. Plans
  - ✓ High availability: replication, data migration
  - ✓ Key-value database cloud
  - ✓ NoSQL(document database), Analytics, …

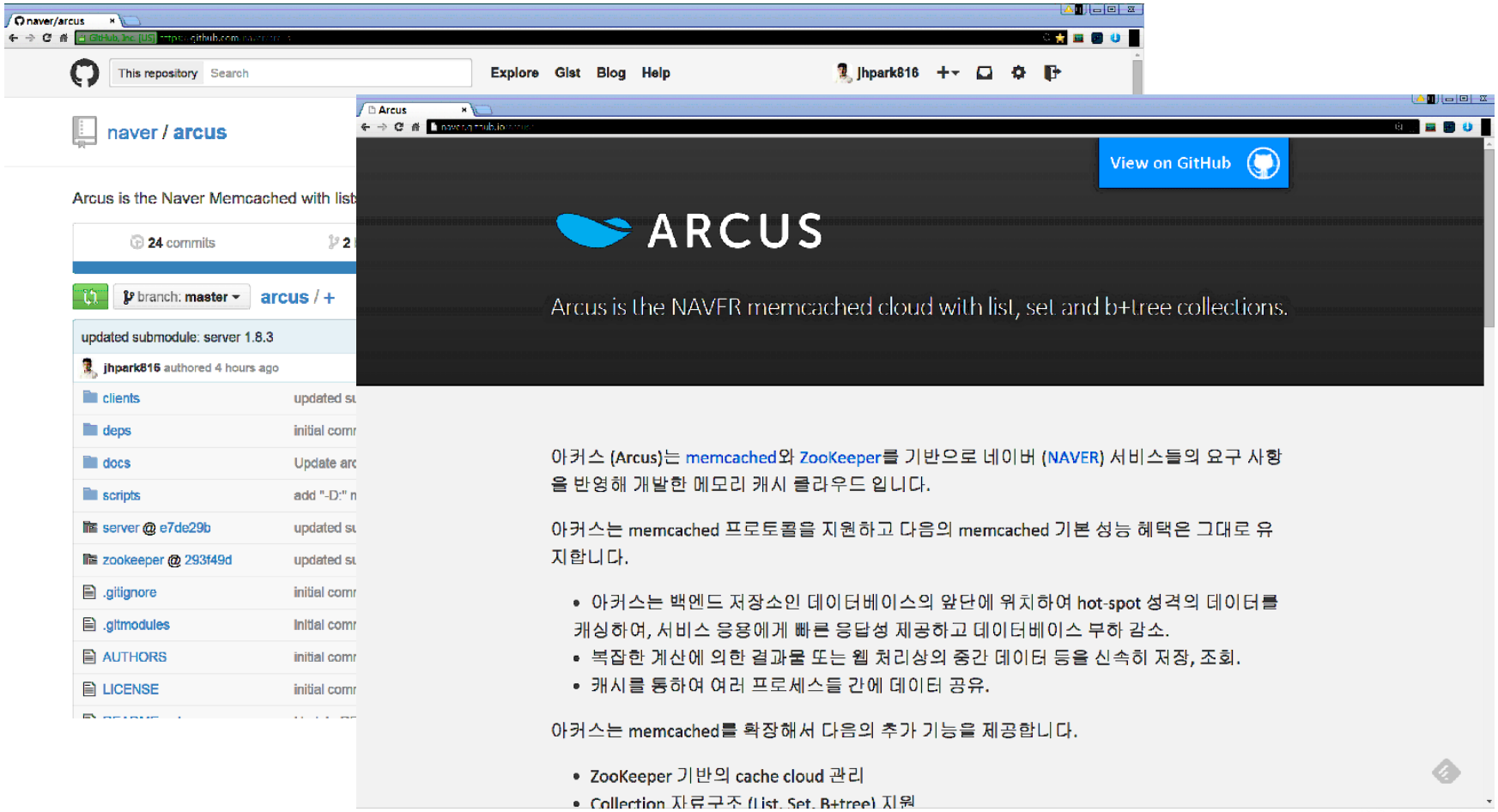
ARCUS CACHE CLOUD

- ARCUS [ɑ́ːrkəs]: 아커스, a kind of cloud
  - ✓ NAVER memcached cloud with list, set, and b+tree collections.

- History
  - ✓ Started to develop ARCUS at NAVER since 2009.
  - ✓ Used in a lot of NAVER services, until now.
  - ✓ Opened to open source SW in May 2014.
    - ✓ Apache License 2.0.
  - ✓ Dev. and support continued by JaM2in since August 2014.

ARCUS CACHE CLOUD

- ARCUS URL – http://naver.github.io/arcus/



What need ARCUS ?

- Services that require high throughput and low latency.
- Services that want to reduce DB query load.
- Services that require data store easy to scale-out.

ARCUS Supported By JaM2in

Who use ARCUS ?

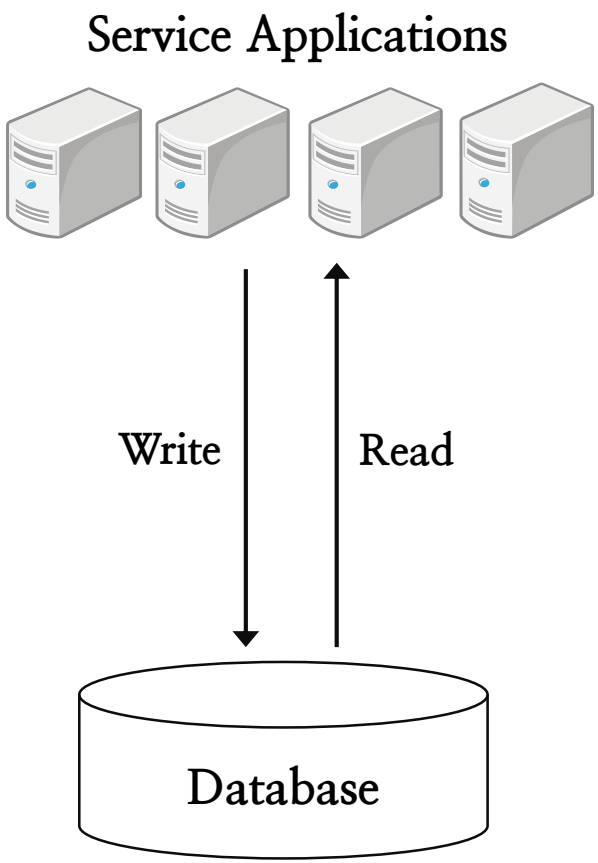
NAVER Me, Café, Blog, Mail, Jisik-iN, Shopping, News, and more

LINE Home, Timeline, Games, and more

BAND

KAKAO Story

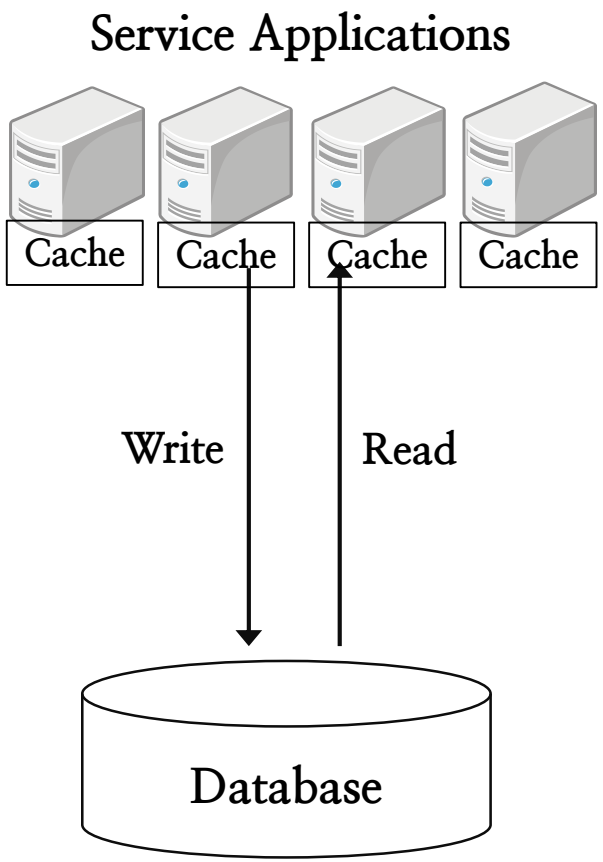
Why ARCUS ?: DB Only



- Large-scale Web Services
  - ✓ Data growth
  - ✓ Increased user requests

- Performance Issues
  - ✓ Low throughput
  - ✓ Slow response
- DB Issues
  - ✓ High cost
  - ✓ Hard to scale-out

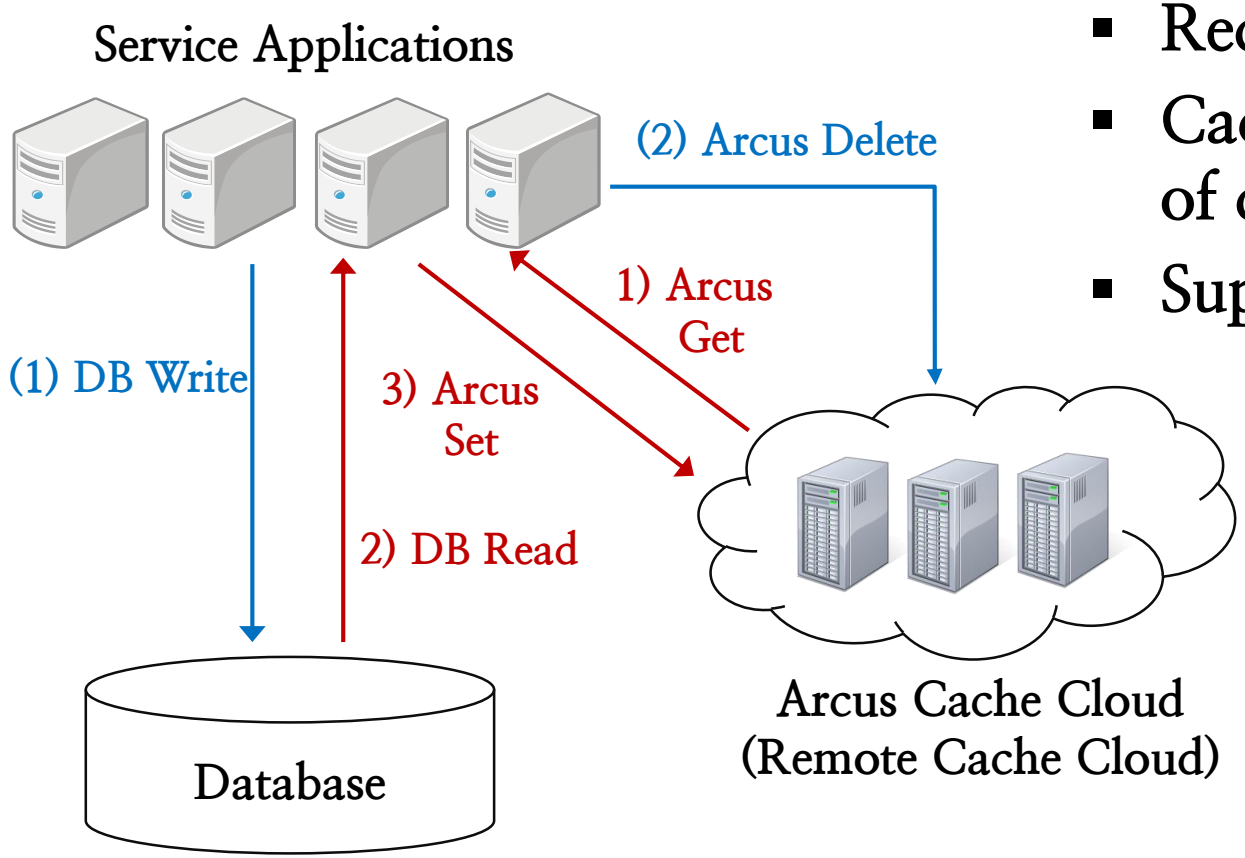
Why ARCUS ?: Local Caching



- Local Caching Issues
  - ✓ Duplicate data
  - ✓ Data inconsistency

- Its use is limited to primitive cache
  - ✓ A small amount of data
  - ✓ Immutable data

Why ARCUS ?: Remote Caching

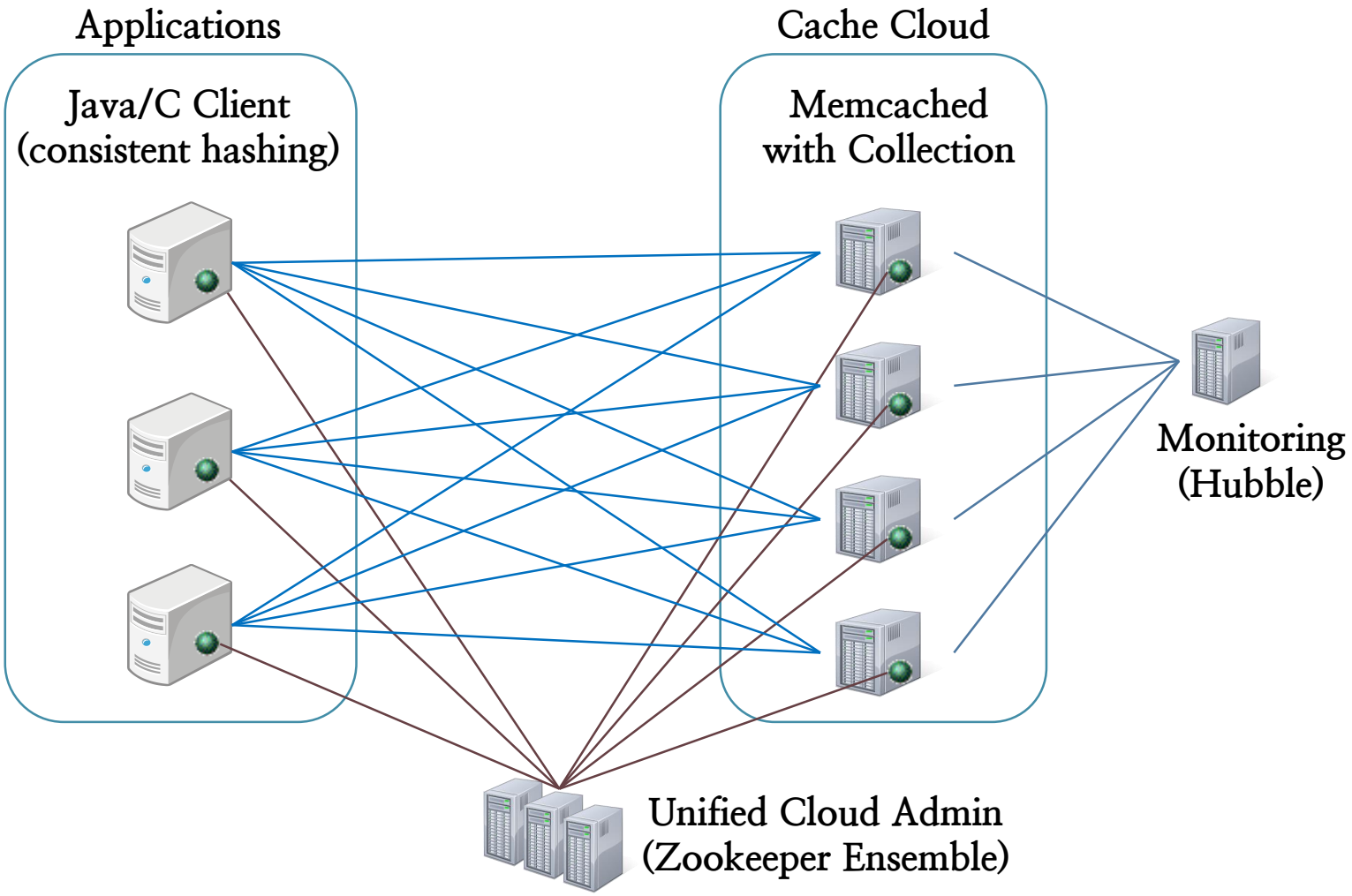


- Ensure high-performance
- Reduce DB load
- Cache large amount of data
- Support Consistency

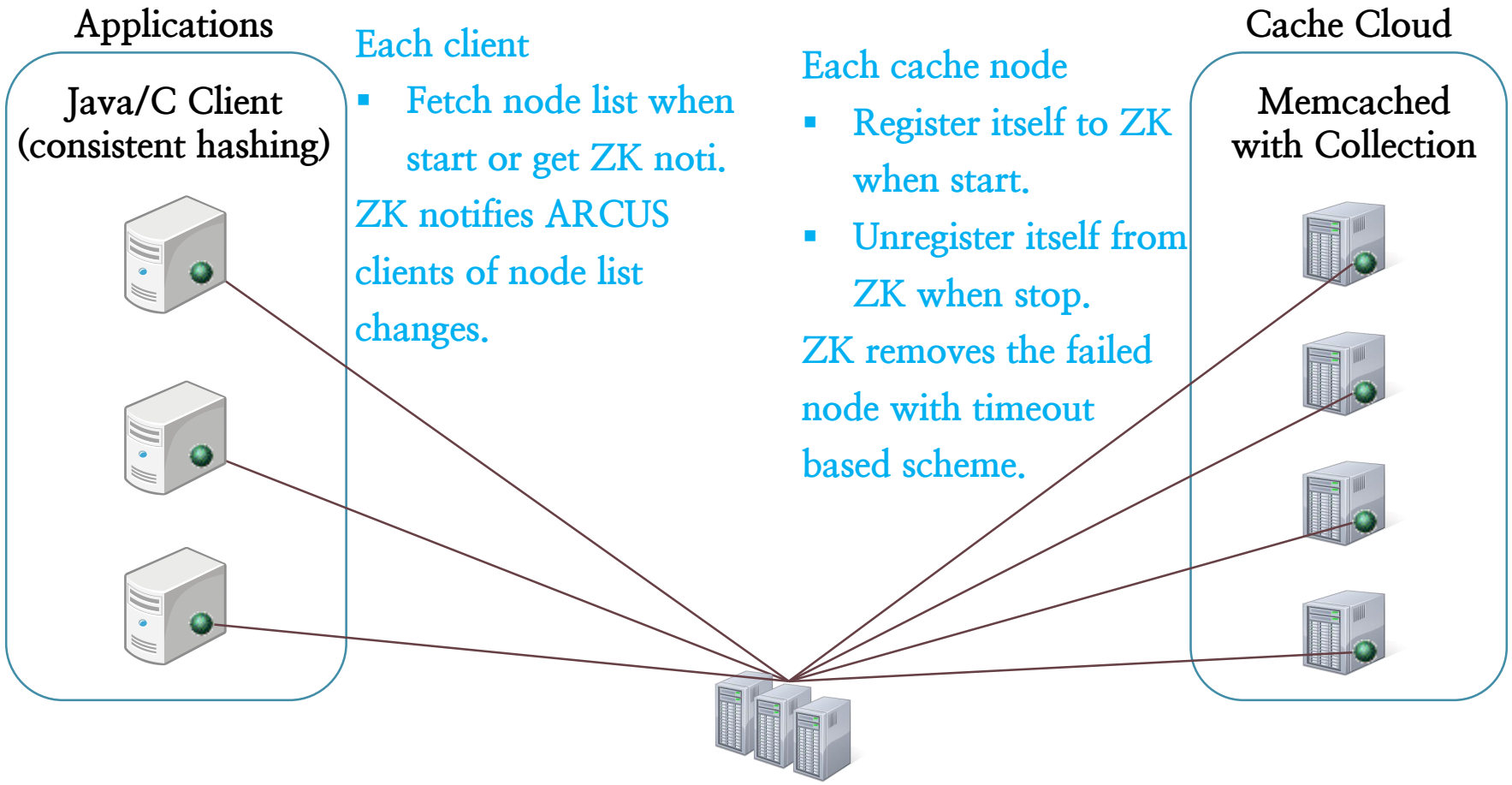
ARCUS Technical Features

- Extended Key-Value data model based on Memcached
  - ✓ Support data collection: List, Set, B+Tree
- High Performance
  - ✓ High throughput of 100K~200K requests/sec (1 node)
  - ✓ Avg. latency of less than 1ms.
- Elastic Cache Cloud based on ZooKeeper
  - ✓ Scale-out, Automatic fail-stop, …
- Other Features
  - ✓ Memory manager optimized for caching
  - ✓ Getting/Setting key-value item attributes
  - ✓ Dynamic configuration settings: maxconns, memlimit, …

ARCUS Architecture

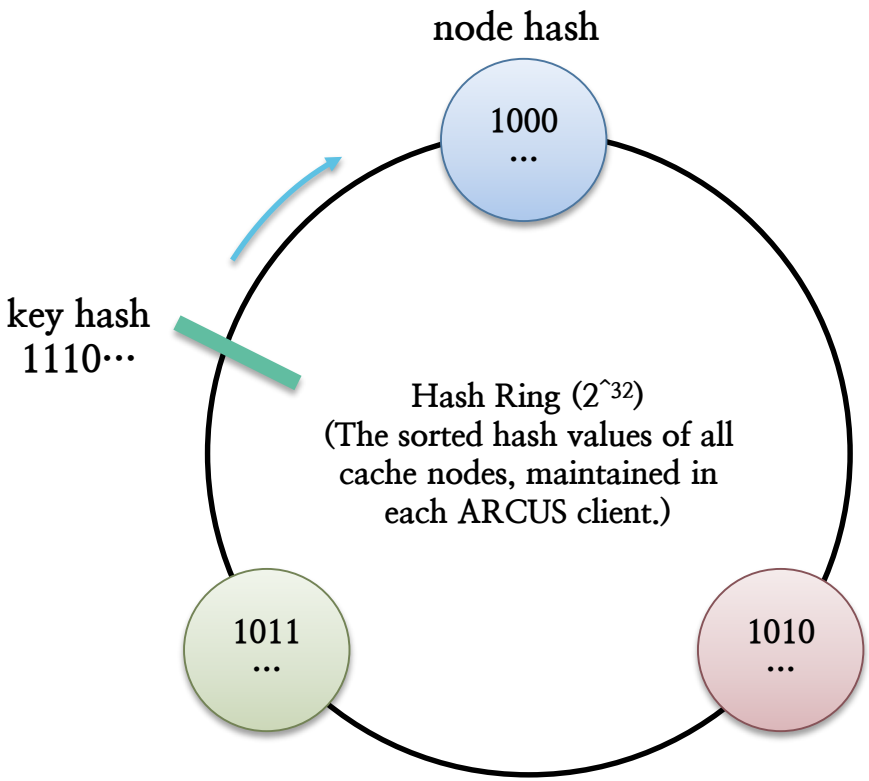


ARCUS Cloud Management



ARCUS Data Distribution

- Consistent Hashing
  1. Construct the hash ring with the hash values of all cache nodes.
  2. For a key, find the first cache node encountered with clock-wise direction from the key hash value.



A cache node is joined/leaved (N: number of cache nodes)

→

Only the cache items in 1/N cache node are re-mapped to other cache nodes.

ARCUS Cache Cloud

- ARCUS Cache Cloud
  - ✓ Distributed memory object caching system
  - ✓ A set of ARCUS cache nodes

- ARCUS Cache Node
  - ✓ Memory object caching node
  - ✓ Hash table : main structure for storing 〈Key, Object〉 items.
  - ✓ Expiration : auto-expiration after the specified time.
  - ✓ Eviction : LRU based eviction in shortage of memory space.

ARCUS Data Model

Key-Value Data Model

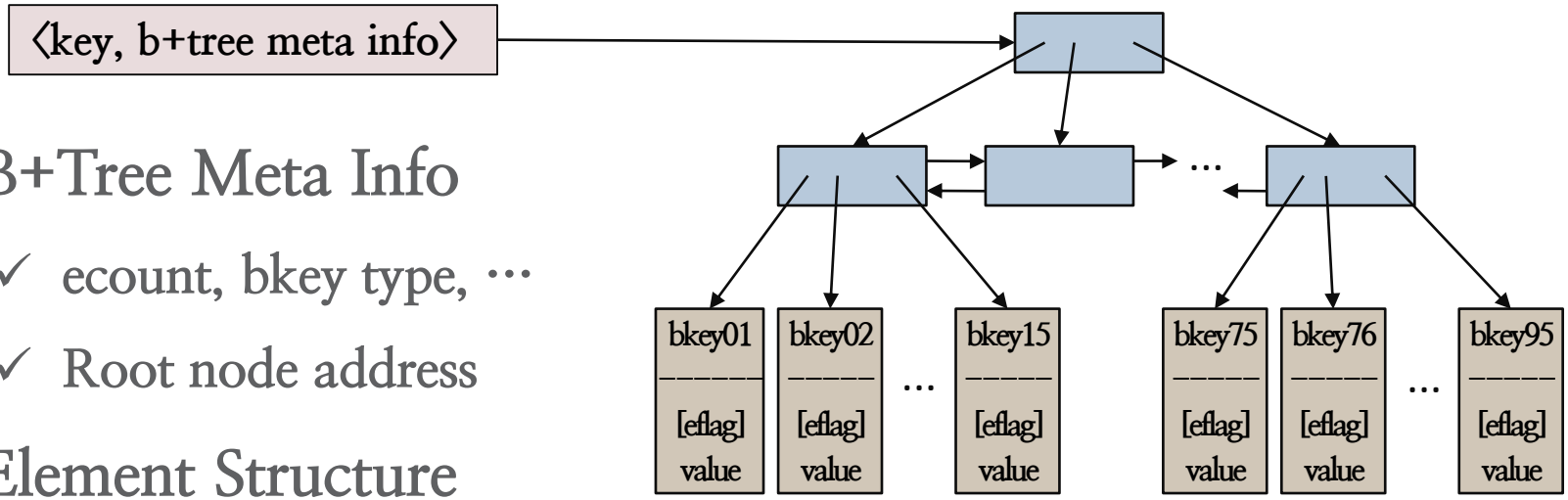
- Key: a key-value item identifier
  - ✓ Format: 〈prefix〉:〈subkey〉 (max 250 characters)
    - ✓ 〈prefix〉: manage a set of items in the logical group.
    - ✓ 〈subkey〉: identify an item in a set of items of the prefix.
- Value: an object stored/retrieved with a key.
  - ✓ Simple key-value item : single value (max 1MB)
  - ✓ Collection item : a collection of values
    - ✓ max 50,000 elements
    - ✓ max 4KB value in each element

ARCUS Collection Type

Type	Features	Use cases in social media services
List	▪ Doubly linked list structure ▪ Access elements with list indexes	
Set	▪ Extendable hash table structure ▪ An unordered set of unique data ✓ Membership checking ▪ Access an element with the value itself.	▪ Store friendships or subscriptions info.
B+Tree	▪ B+tree structure ▪ An ordered data set based on b+tree key ▪ Access elements with bkey(b+tree key) ▪ Access elements with b+tree position.	▪ Store the post id list of friends in reverse time order. ▪ Fetch the latest N post ids of friends.

ARCUS B+Tree Structrue

- B+Tree Item Structure



- B+Tree Meta Info
  - ✓ ecount, bkey type, …
  - ✓ Root node address
- Element Structure

bkey	▪ B+tree key, an 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
value	▪ Data field stored/retrieved together with bkey. (max 4KB)

ARCUS B+Tree Get

- B+Tree get
  - ✓ 〈key〉 : 〈bkey_range, [eflag_filter,] [[offset,] count]〉
  - ✓ ==〉 a set of elements

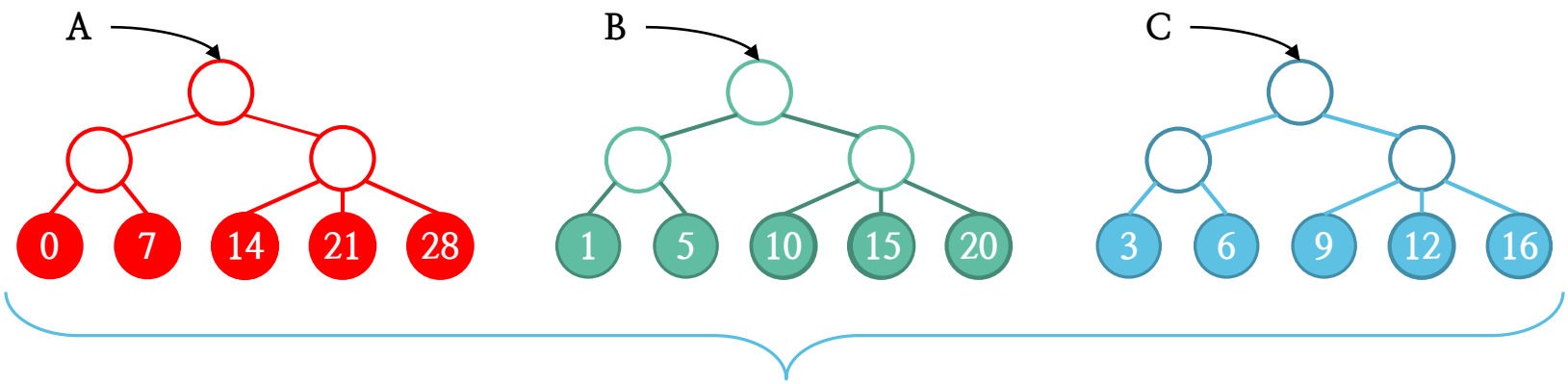
Condition	Description
bkey_range	▪ Mandatory, ascending or descending order ▪ Ex) 100..200, 200..100, 0x00AA..0x00FF
[eflag_filter]	▪ Optional filter condition applied to the value of eflag. ▪ [bitwise operator +] comparison operator ✓ bitwise : AND, OR, XOR ✓ comparison: EQ, NE, LT, LE, GT, GE, IN, NOT IN
[[offset,] count]	▪ Optional skip and retrieval count

ARCUS B+Tree Position Operations

- B+Tree find position
  - ✓ 〈key〉 : 〈bkey, order〉 ==〉 a position
    - ✓ 〈order〉: ASC or DESC
- B+Tree get by position
  - ✓ 〈key〉 : 〈order, position_range〉 ==〉 a set of elements
- B+Tree find position with get
  - ✓ 〈key〉 : 〈bkey, order, count〉 ==〉 a set of 〈position, element〉 pairs

ARCUS B+Tree Sort-Merge Get Operation

- B+Tree smget
  - ✓ 〈key_list〉 : 〈bkey_range, [eflag_filter,] [[offset,] count]〉
  - ✓ ==〉 a set of elements



Get elements with 30 〉= bkey 〉= 10 from A, B, C

⇩

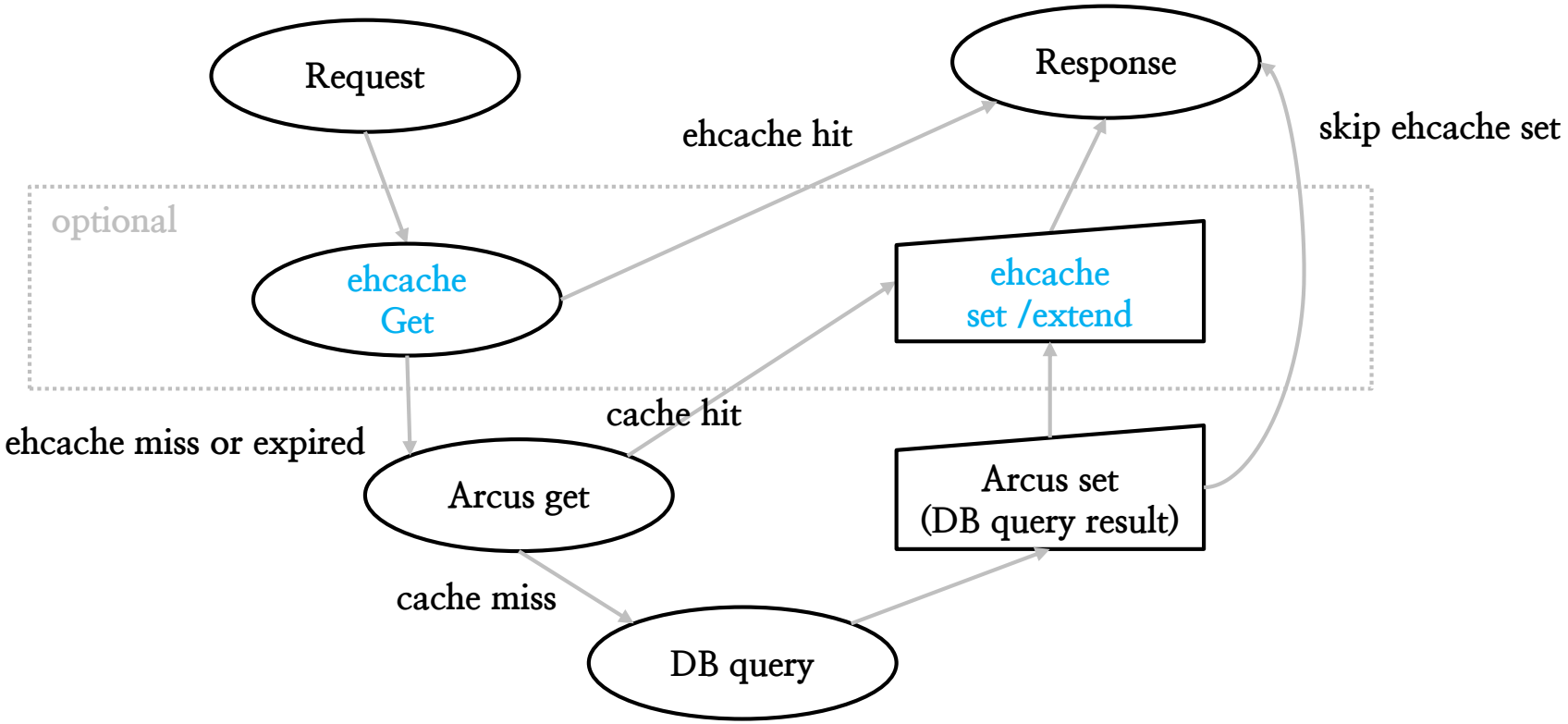
[28, 21, 20, 16, 15, 14, 12, 10]

ARCUS Operation List

Operation Type		Operation List
Simple Key-Value		get, set, add, replace, delete, incr/decr, …
List Collection	Item	create, drop
	Element	insert, delete, get
Set Collection	Item	create, drop
	Element	insert, delete, get, exist
B+Tree Collection	Item	create, drop
	Element	insert/upsert, update, delete, get, count, incr/decr, mget, smget, position, gbp, pwg
Other Operations		getattr, setattr, flush, stats, config, …

ARCUS Client

- Data compression in java client
- Front caching in java client



ARCUS Misc.

- OS – Linux Only
  - ✓ CentOS 64bit – Fully tested
  - ✓ Redhat/Ubuntu 64bit – Partially tested

- Clients Provided Officially
  - ✓ Java, C

Part 2. ARCUS Use Cases

LINE Home/Timeline

- Home : Select posts that a certain friend can view.
- DB Issue : very difficult to use a general DBMS

```
//  flags: 2^0=Family, 2^1=School, 2^2=Tennis, 2^3=Work, …
// 1. Select posts that any friend can view
SELECT * FROM post WHERE flags = 0;
// 2. Select posts that school and work friends can view
SELECT * FROM post WHERE (flags & (2^1 + 2^3)) or flags=0;
```

- How to use ARCUS ?
  - ✓ B+tree : maintain the postID list with group-permit per post
  - ✓ Get the permitted postID list with eflag filtering on elements.
- Source - LINE Social Network Service Architecture (2014/06)
  - ✓ http://d2.naver.com/helloworld/809802

LINE Games

- Requirements
  - ✓ View the ranking of an user score.
  - ✓ View N 〈score, user〉 pairs before and behind a score.
- How to use ARCUS ?
  - ✓ How to store top game scores ? B+Tree
    - ✓ 〈bkey: score, data: userinfo〉
  - ✓ Request Case 1)
    - ✓ Find a position with a 〈score, order(DESC)〉 pair.
    - ✓ Find 〈score, userinfo〉 pairs with the position range or score range.
  - ✓ Request Case 2)
    - ✓ Find a position and 〈score, userinfo〉 pairs with score, order, count.

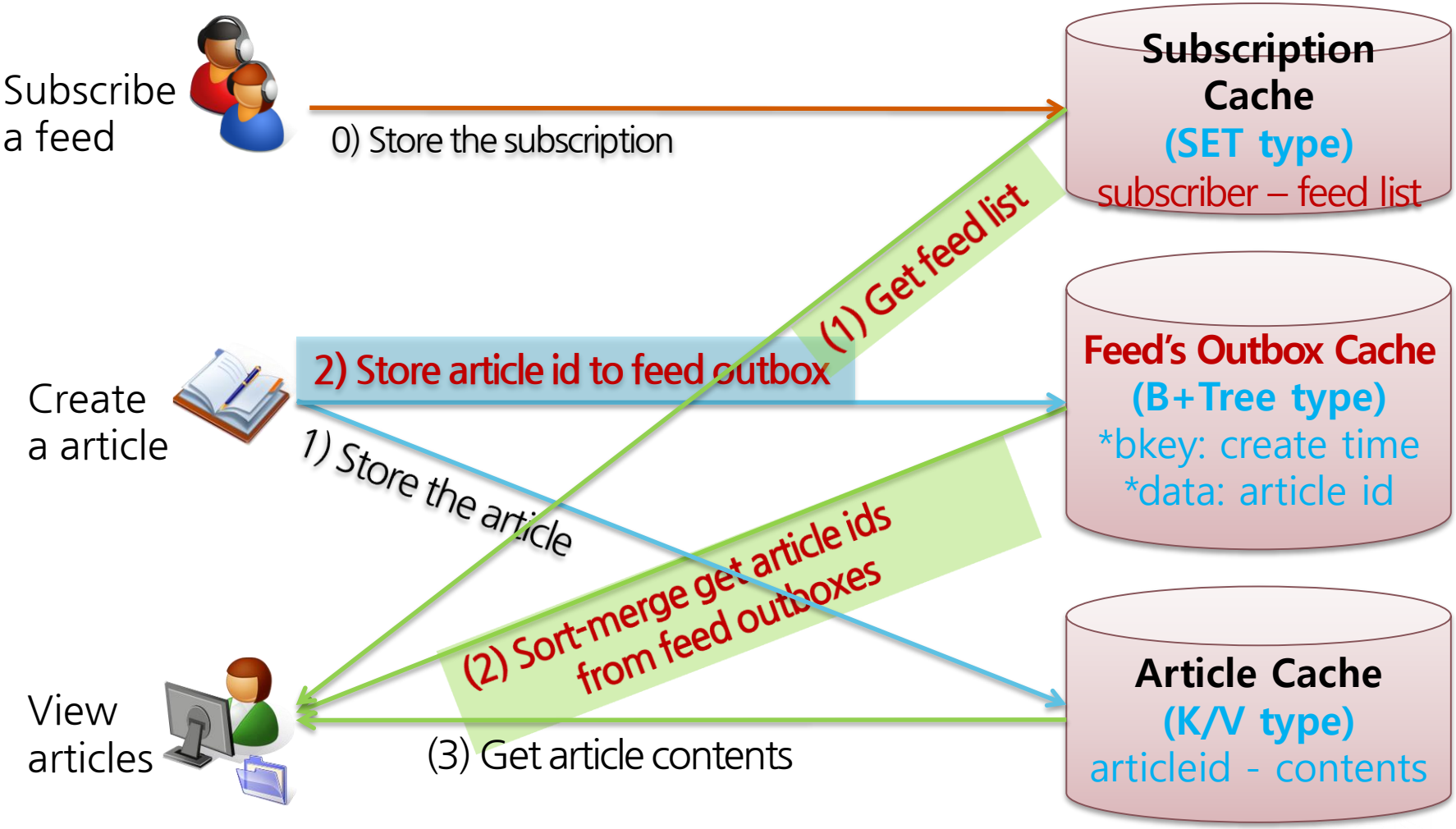
NAVER Me (1)

DB: view the latest 20 articles (large feed list => too slow)

```
SELECT * FROM articles
WHERE feedid in (feedID1, feedID2, …, feedIDn) AND create_time < sysdate()
ORDER BY create_time DESC
LIMIT 20;
```

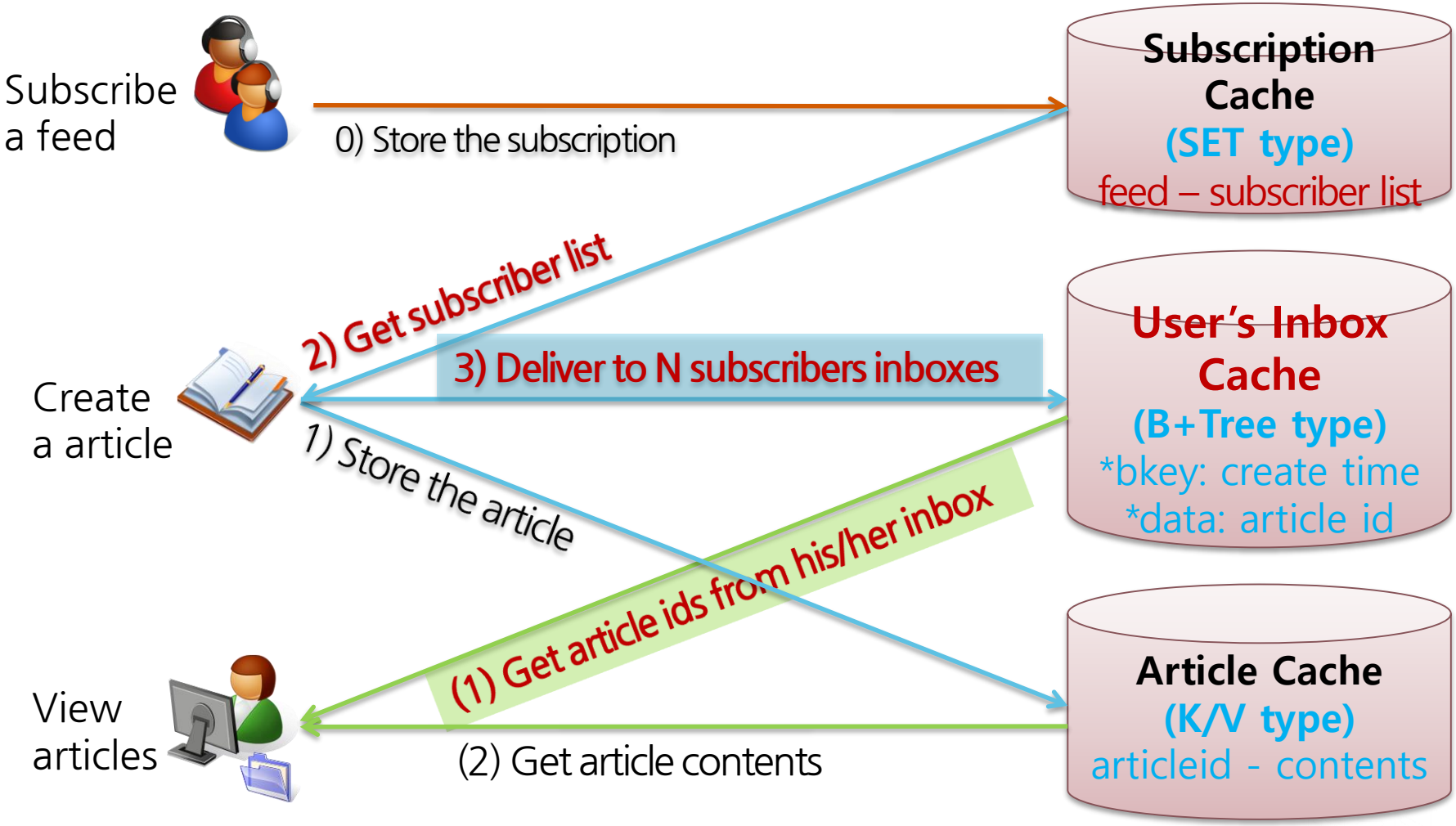
NAVER Me (2)

Push(user inbox) vs. Pull(feed outbox & sort-merge get)



NAVER Me (3)

Push(user inbox) vs. Pull(feed outbox & sort-merge get)



KAKAO Story

- How to use ARCUS ?
  - ✓ Store user profiles, posts, friendships.
  - ✓ Detailed use case is not published.
- Expected to be similar with NAVER Me case.

Part 3. ARCUS Open Source

ARCUS Open Source

- Opened Sources (opened by Naver)
  - ✓ ARCUS cache server & Java/C clients
  - ✓ Zookeeper library with Arcus modification
  - ✓ ARCUS monitoring system (Hubble)
- Our contributions
  - ✓ The first open source cache solution in Korea
  - ✓ Soon, release the advanced ARCUS with high availability: replication, data migration
  - ✓ Good technical partner with large-scale web service providers

To be Expected…

- Contribute to develop various services and to improve service quality
- Educate & train open source SW developers, especially skilled in data technology
- Contribute to win-win partnership between (open source) system SW and service development

Thank You !!