Greenland IMDG

off-heap 메모리를 이용한 data structure 기반의 IMDG 개발

송원근 장태영 한민형

목차

개발 목표

Off-heap 메모리를 구현하여

Full GC소모시간을 없애고

빅데이터를 빠르게 다루기 위한 자료구조가 내장된

In-Memory Data Grid 개발

개발 목표

150MB/S

800X

500MB/S

40X

20,000MB/S

PCI EXPRESS®

3,000MB/S

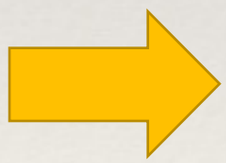
7X

개발 목표

전형적인 OLTP(OnLine Transaction Processing): 1TB

x64 아키텍처 메모리 최대 지원량: 8TB

19인치 블레이드 서버의 평균 메모리: 1TB



메인 메모리의 용량이 저장소의 용량과 유사해져서
IMDG의 필요성이 대두되기 시작했다

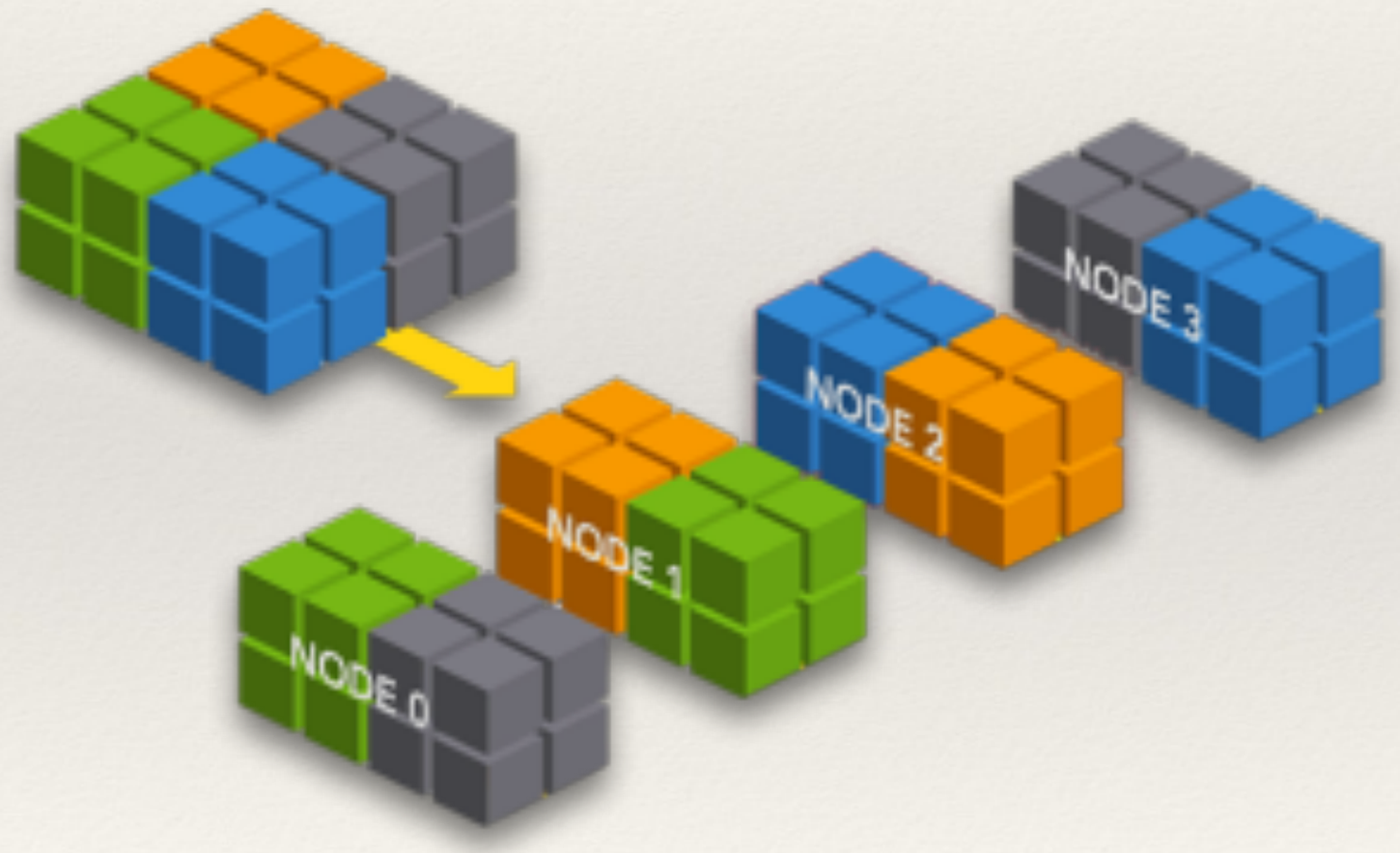
개발 항목

- Java를 통한 IMDG(In Memory Data Grid)개발
- Off-heap 메모리를 구현하기 위해 direct buffer를 사용한 Memory Allocator 개발
- Data Structure를 기반으로 한 인터페이스 제공
- Java에서 IMDG에 접근하고 이용할 수 있는 라이브러리와 Client 개발
- IMDG의 모니터링과 관리를 위한 웹 콘솔 개발

개발 내용

- Java를 통한 IMDG(In Memory Data Grid)개발

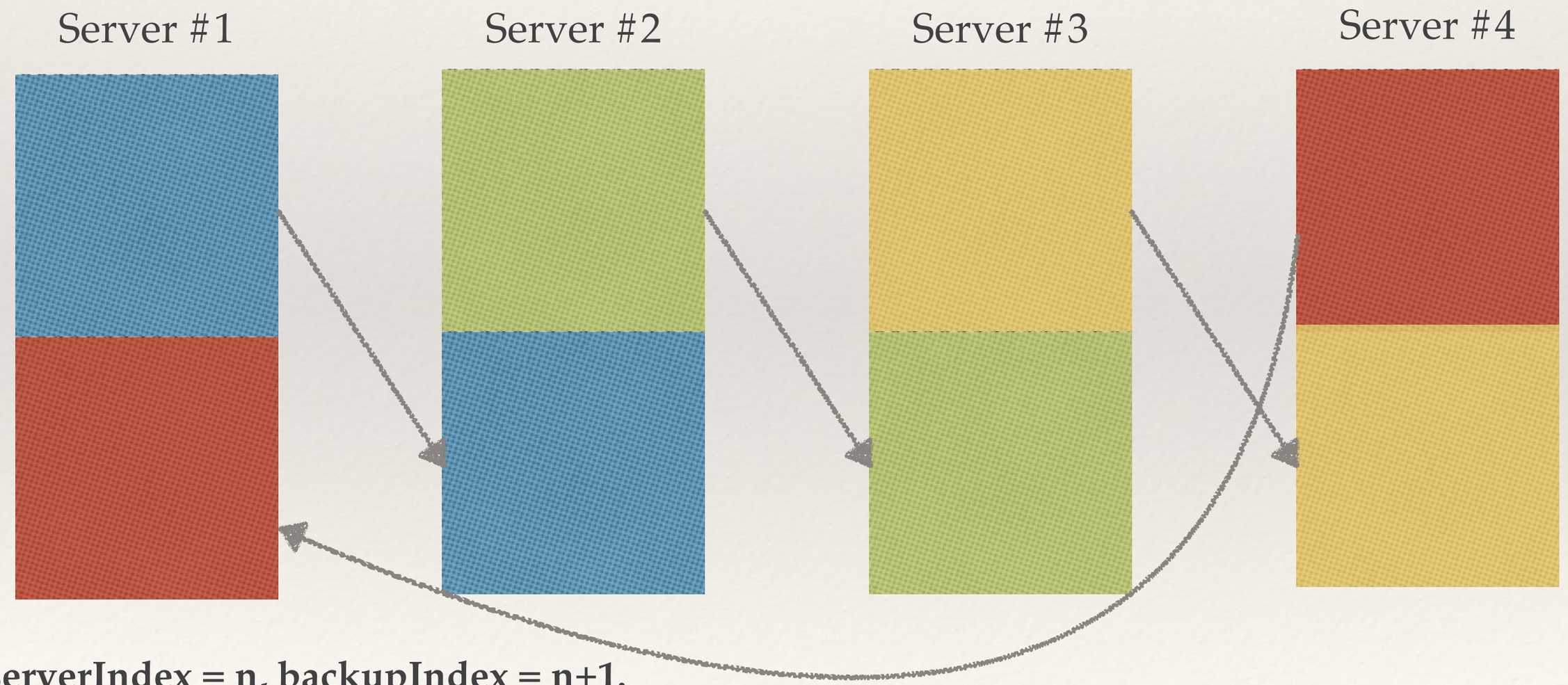
Web형 네트워크로 구현하여 각각의 인스턴스에 대해 사용자의 요청을 처리할 수 있게 설계



개발 내용

- Java를 통한 IMDG(In Memory Data Grid)개발

실시간 타 서버에 백업 프로세스를 적용하여 데이터의 유실을 방지

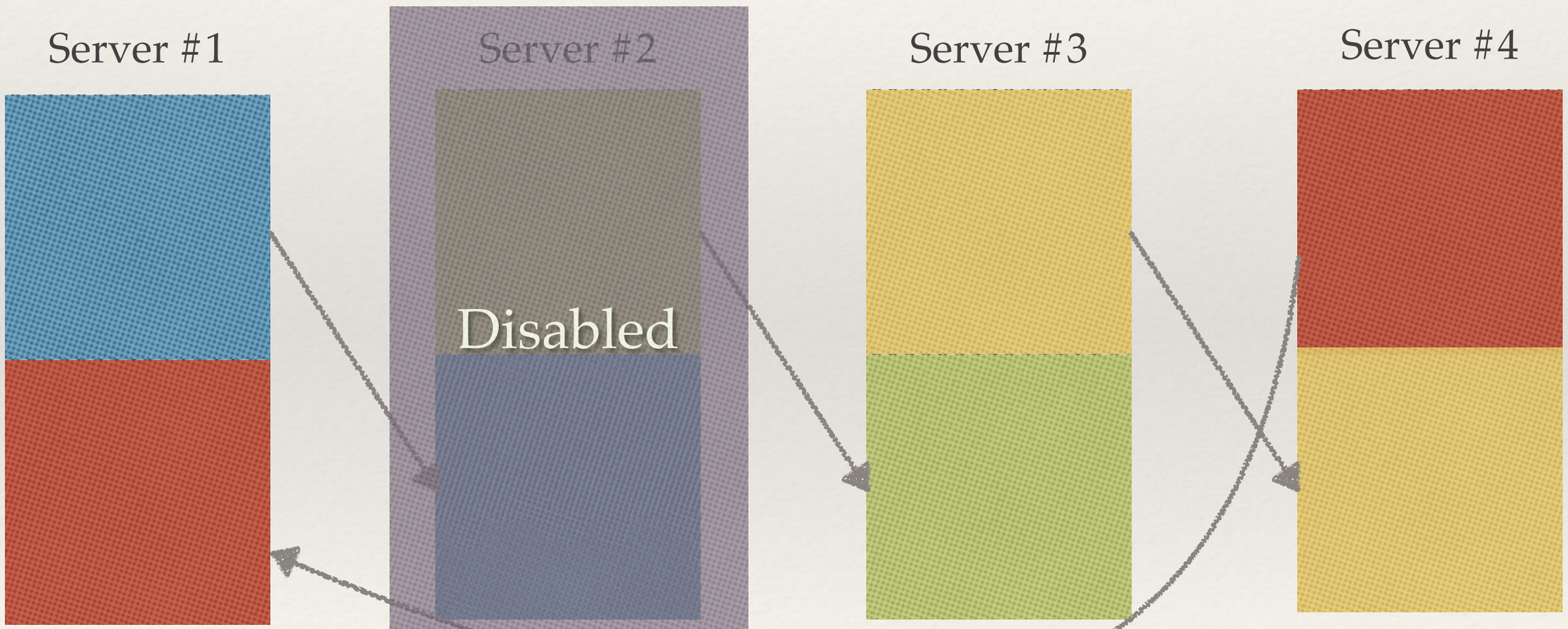


if serverIndex = n, backupIndex = n+1.
if backupIndex >= serverCount, backupIndex = backupIndex - serverCount

개발 내용

- Java를 통한 IMDG(In Memory Data Grid)개발

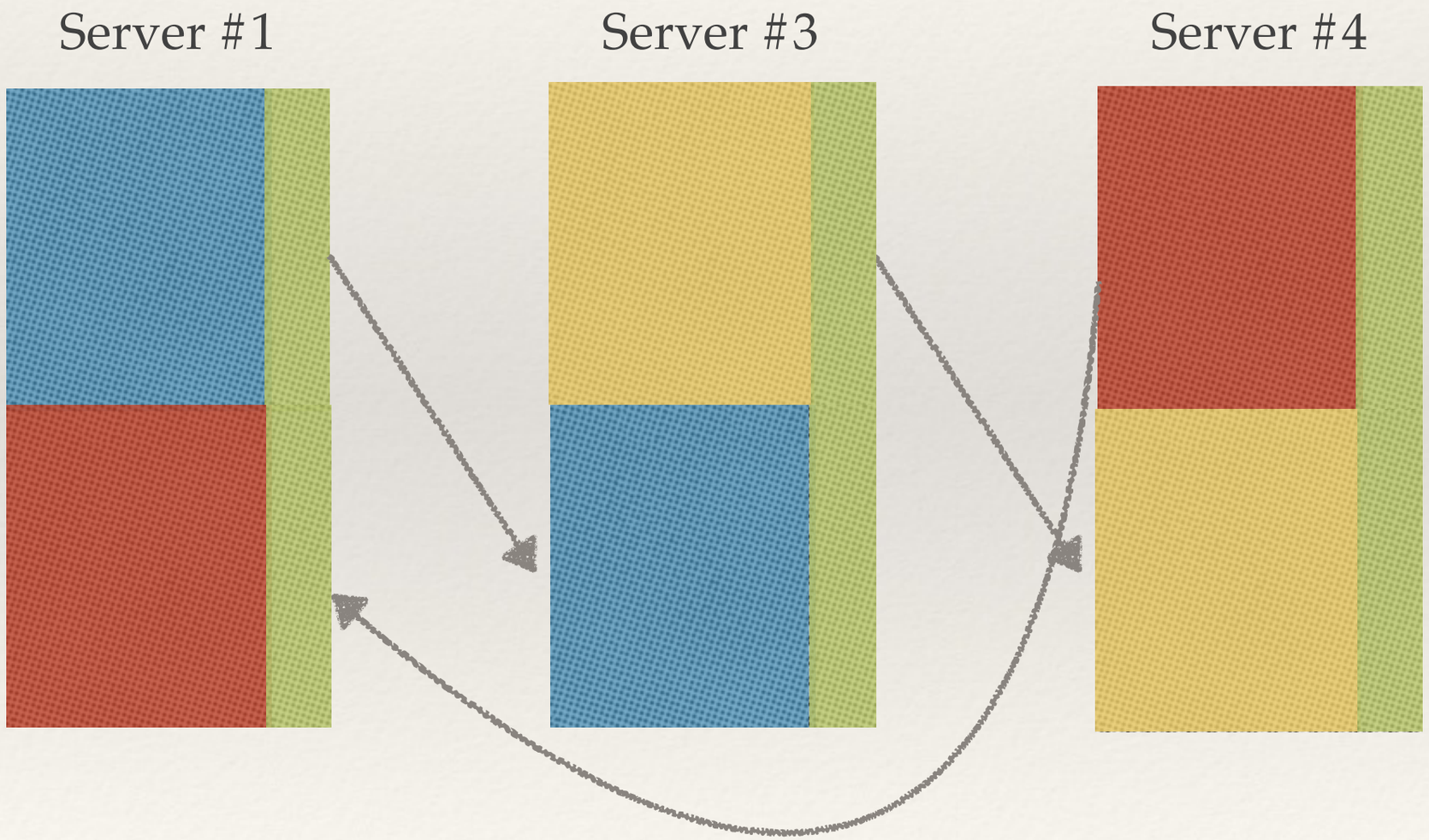
서버가 삭제되거나 추가될 경우, 재 분배 프로세스를 통해 데이터를 사용할 수 있게 한다



개발 내용

- Java를 통한 IMDG(In Memory Data Grid)개발

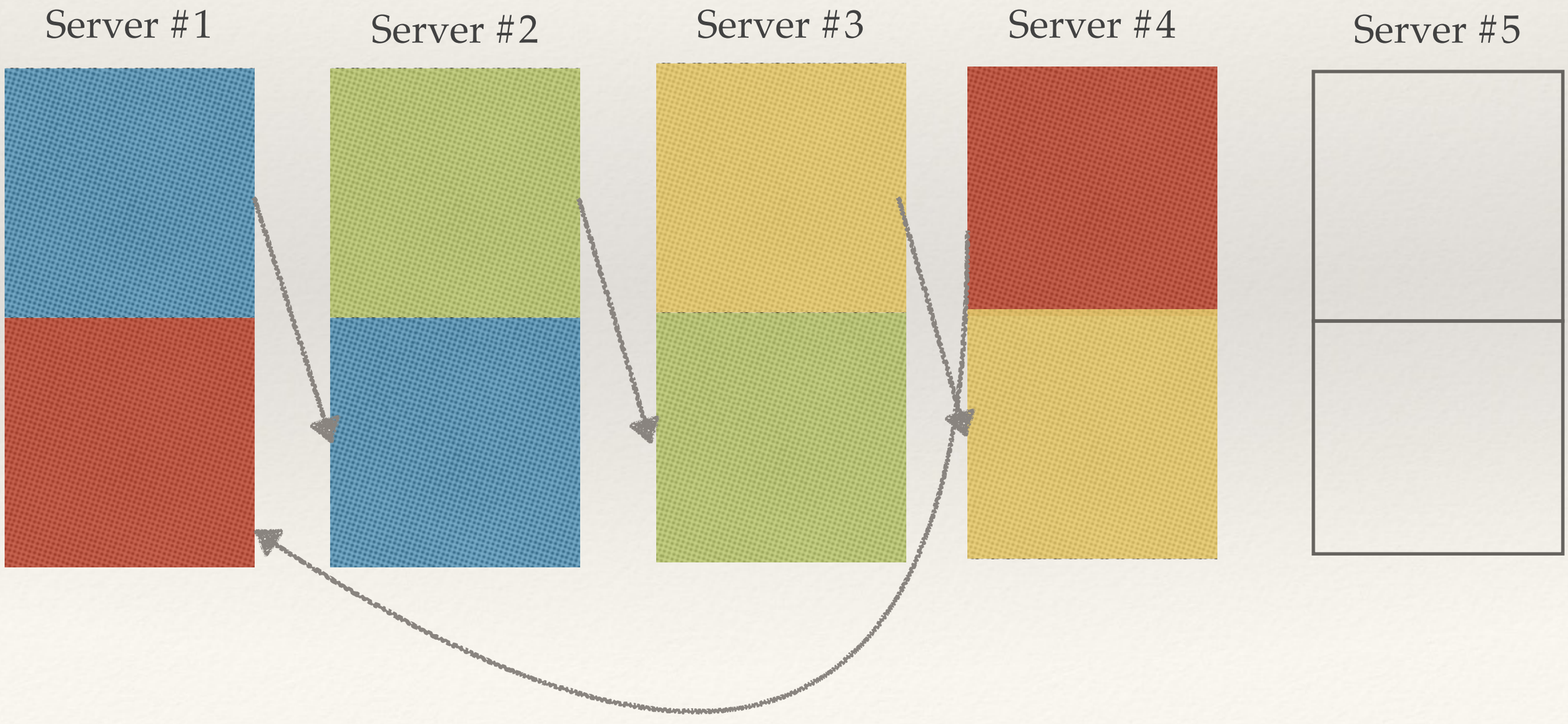
서버가 삭제되거나 추가될 경우, 재 분배 프로세스를 통해 데이터를 사용할 수 있게 한다



개발 내용

- Java를 통한 IMDG(In Memory Data Grid)개발

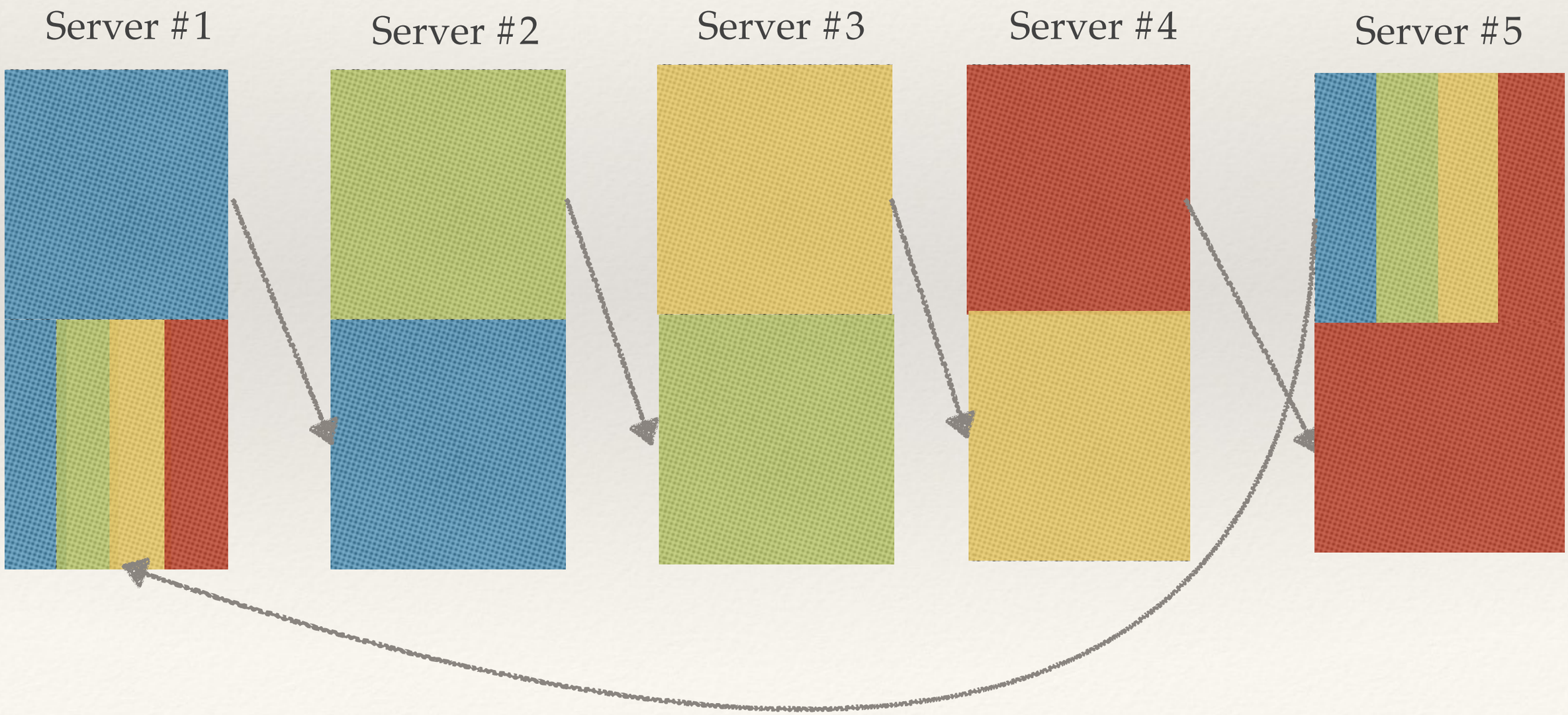
서버가 삭제되거나 추가될 경우, 재 분배 프로세스를 통해 데이터를 사용할 수 있게 한다



개발 내용

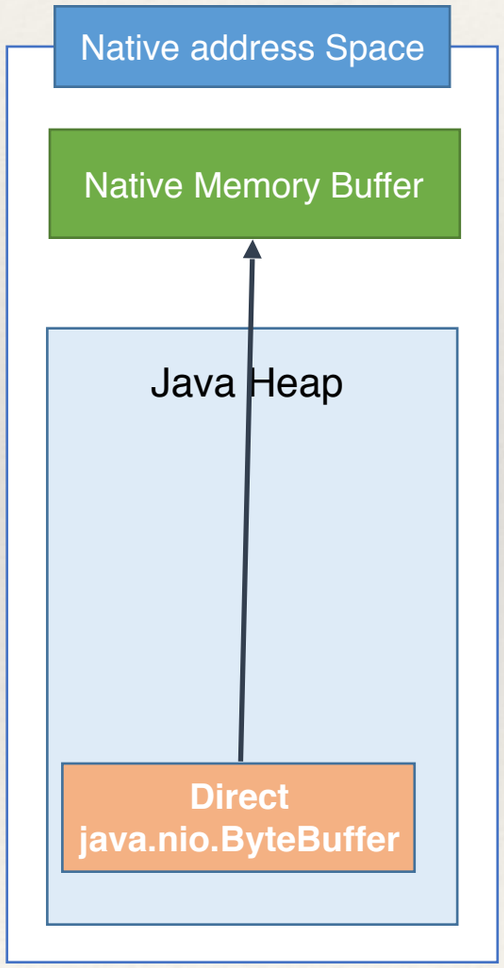
- Java를 통한 IMDG(In Memory Data Grid)개발

서버가 삭제되거나 추가될 경우, 재 분배 프로세스를 통해 데이터를 사용할 수 있게 한다

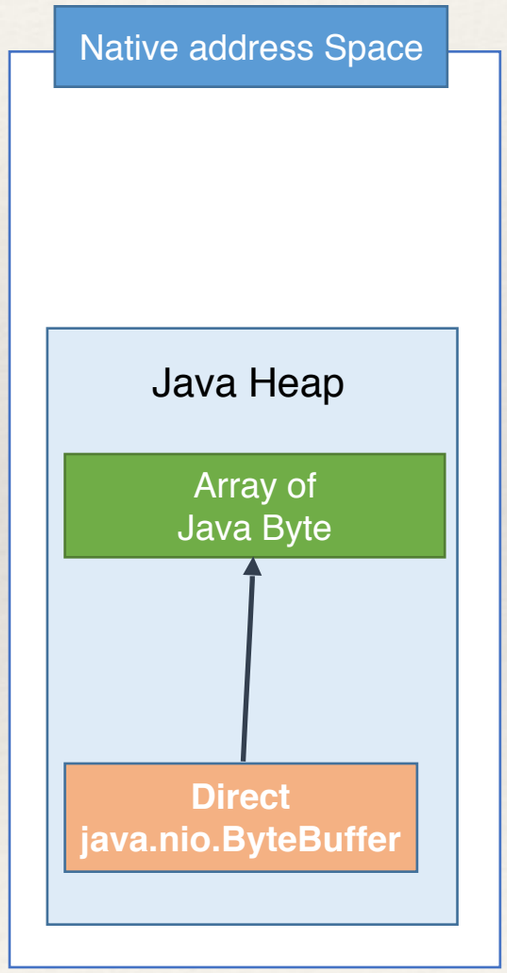


개발 내용

- Off-heap 메모리를 구현하기 위해 direct buffer를 사용한 Memory Allocator 개발



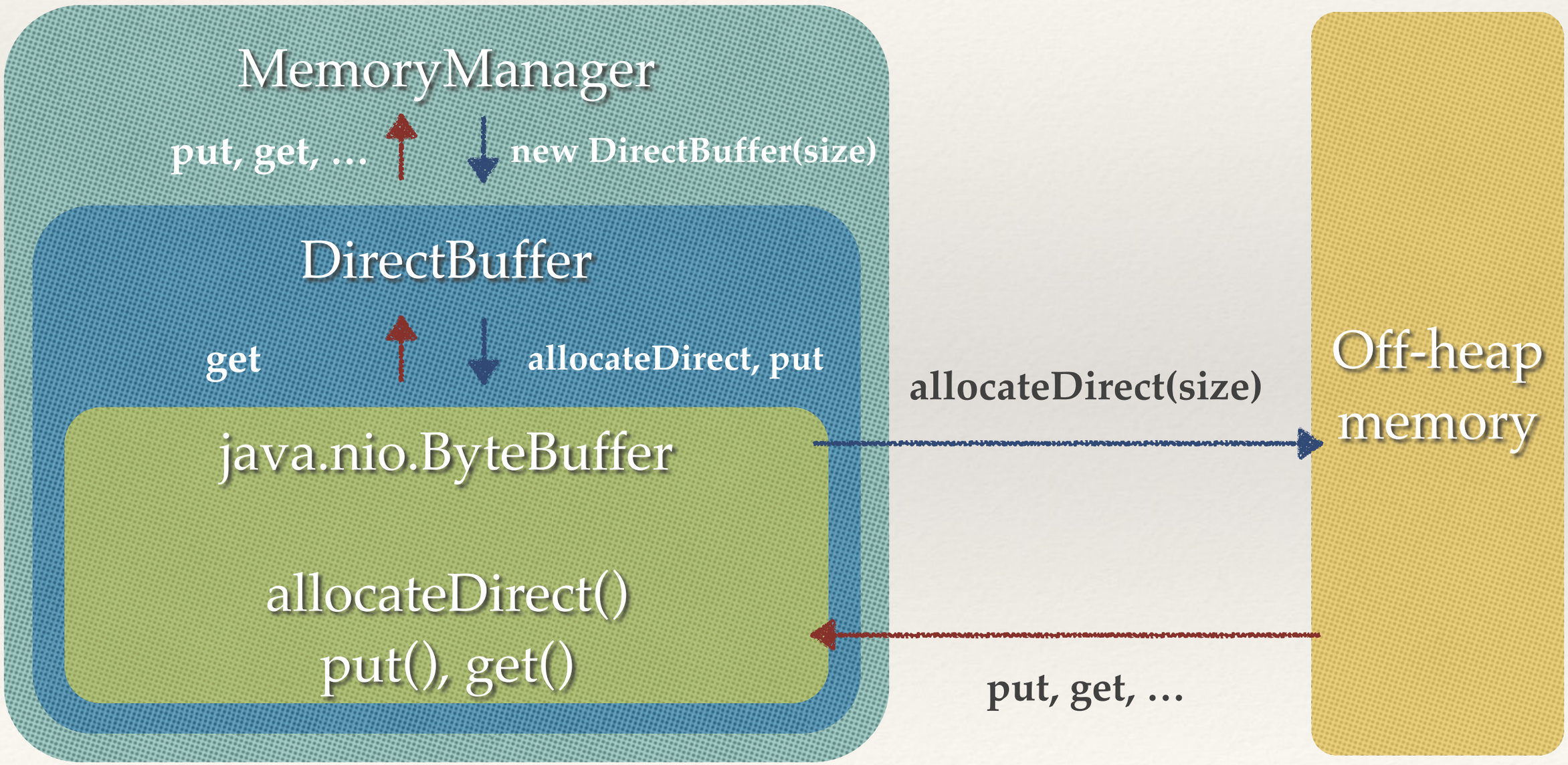
Off-heap 위상 전략



On-heap 위상 전략

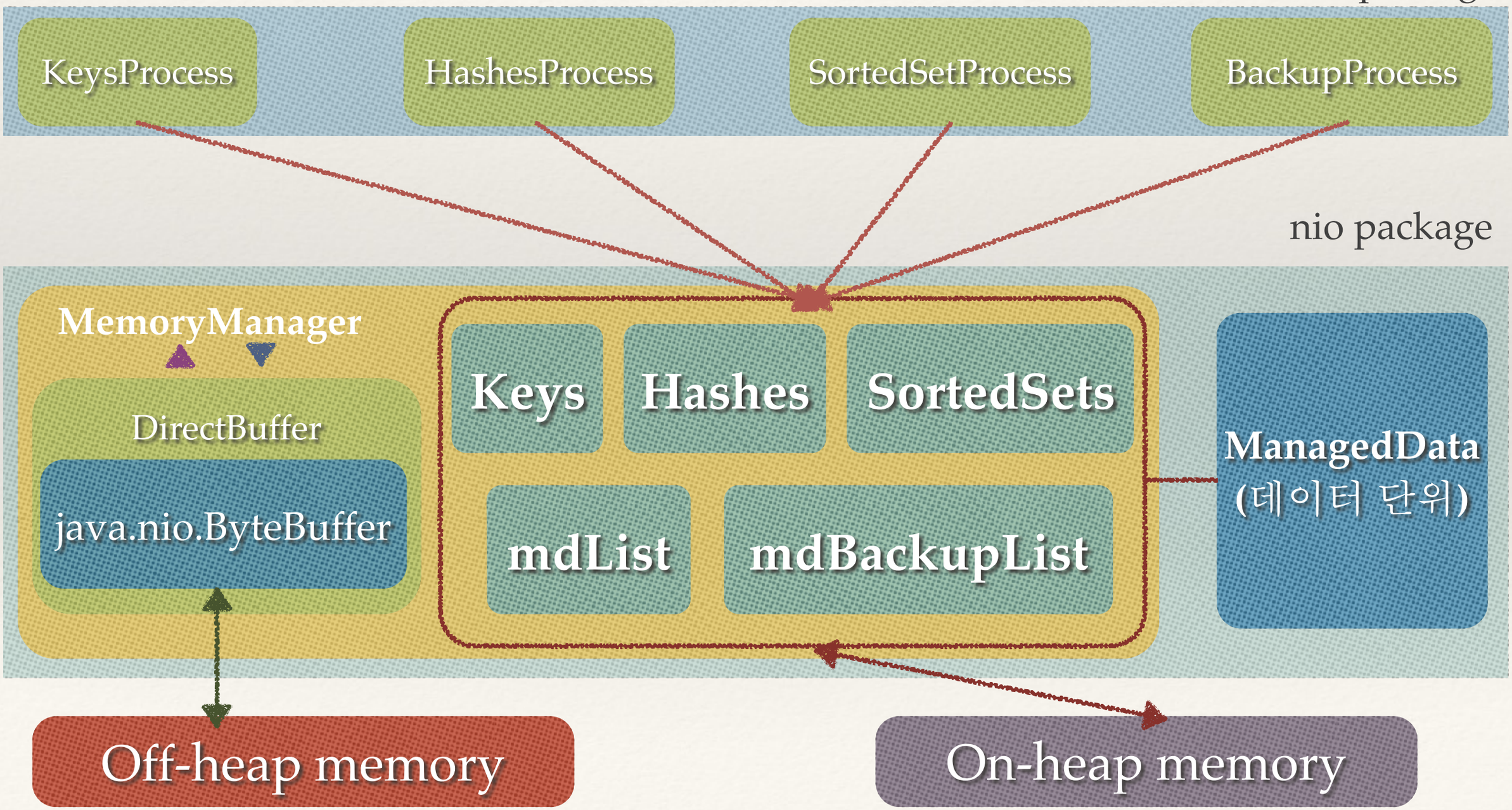
개발 내용

- Off-heap 메모리를 구현하기 위해 direct buffer를 사용한 Memory Allocator 개발



개발 내용

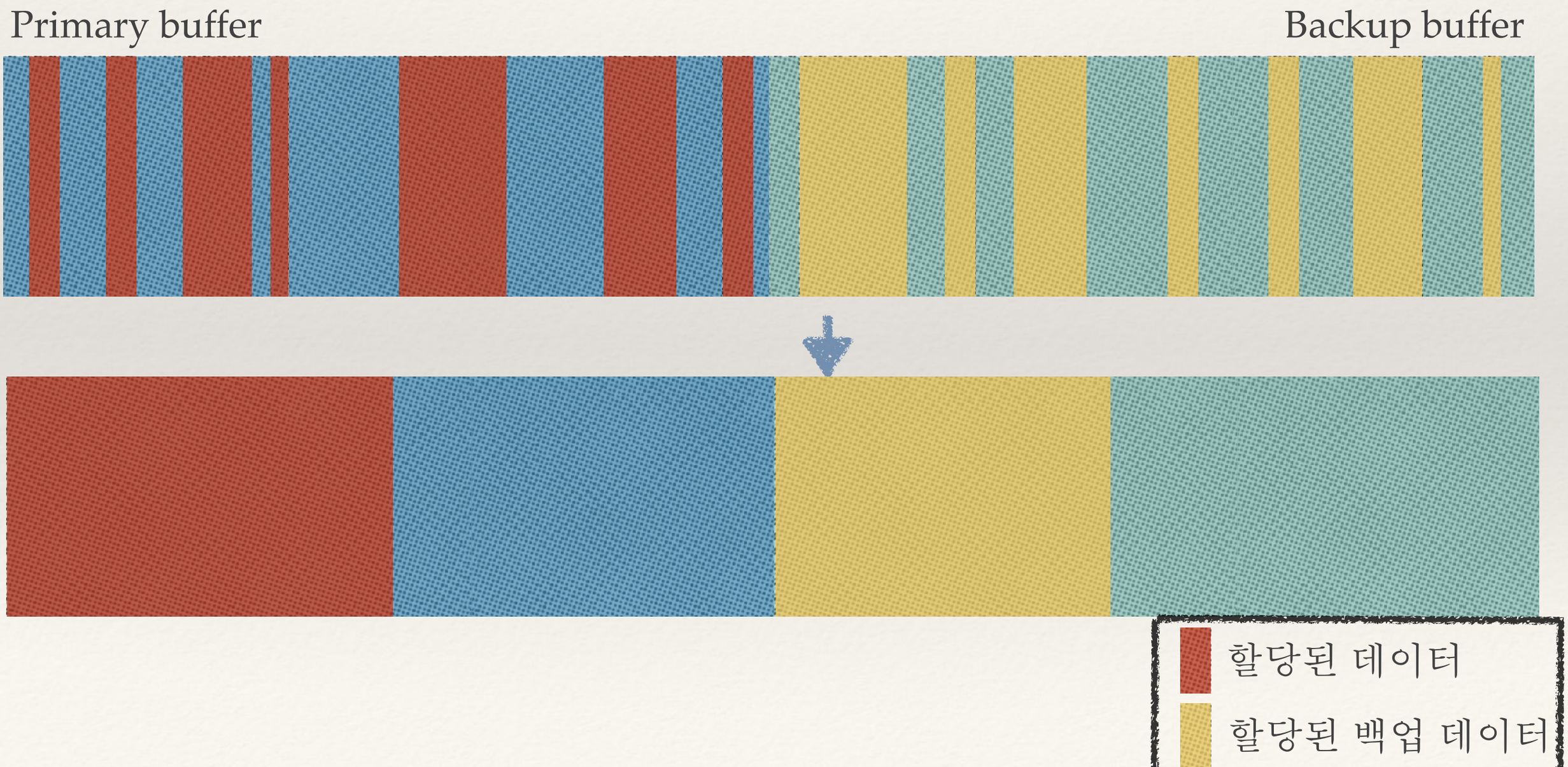
- Off-heap 메모리를 구현하기 위해 direct buffer를 사용한 Memory Allocator 개발



개발 내용

- Off-heap 메모리를 구현하기 위해 direct buffer를 사용한 Memory Allocator 개발

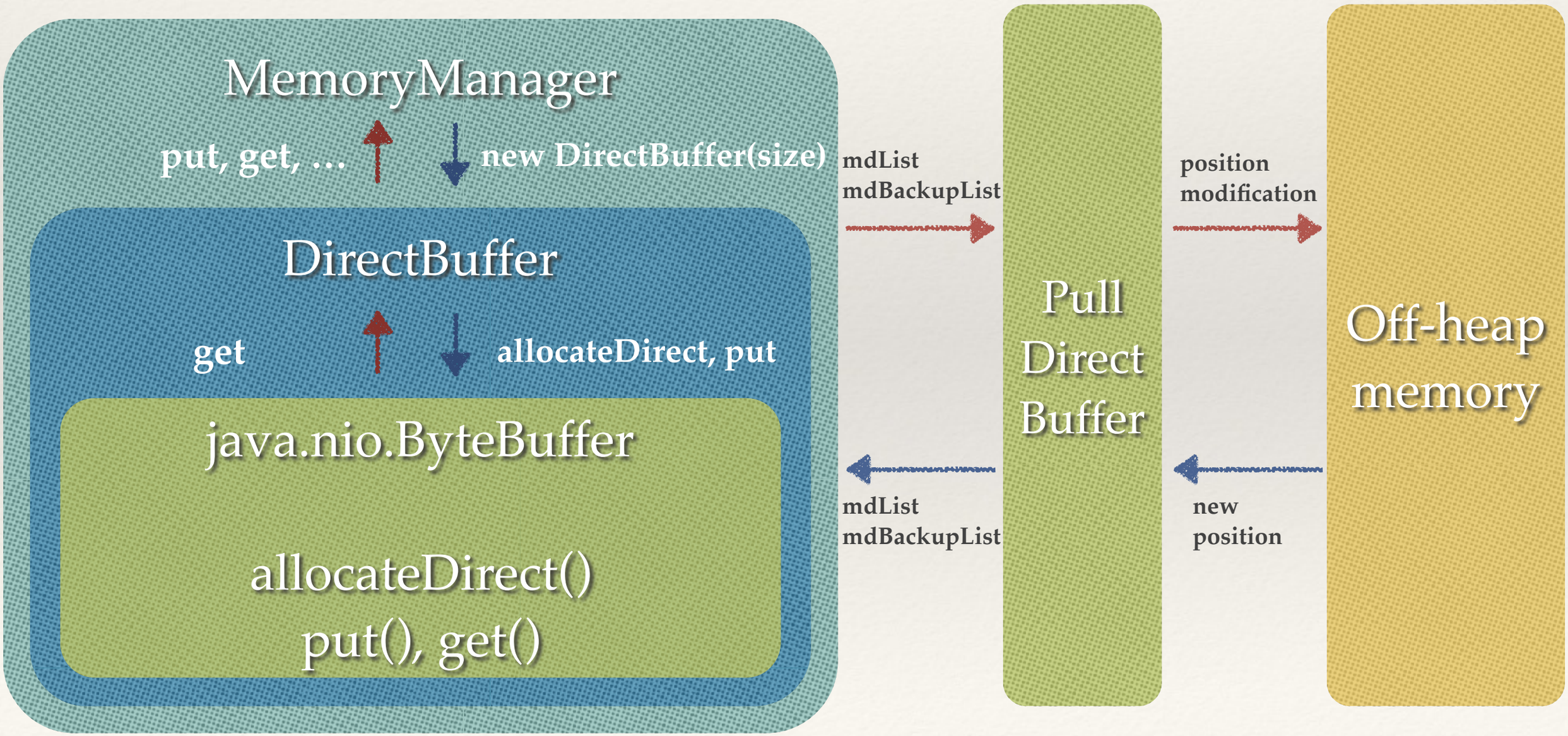
DirectBuffer의 단편화를 해결하기 위한 **PullDirectBuffer** 클래스를 개발



개발 내용

- Off-heap 메모리를 구현하기 위해 direct buffer를 사용한 Memory Allocator 개발

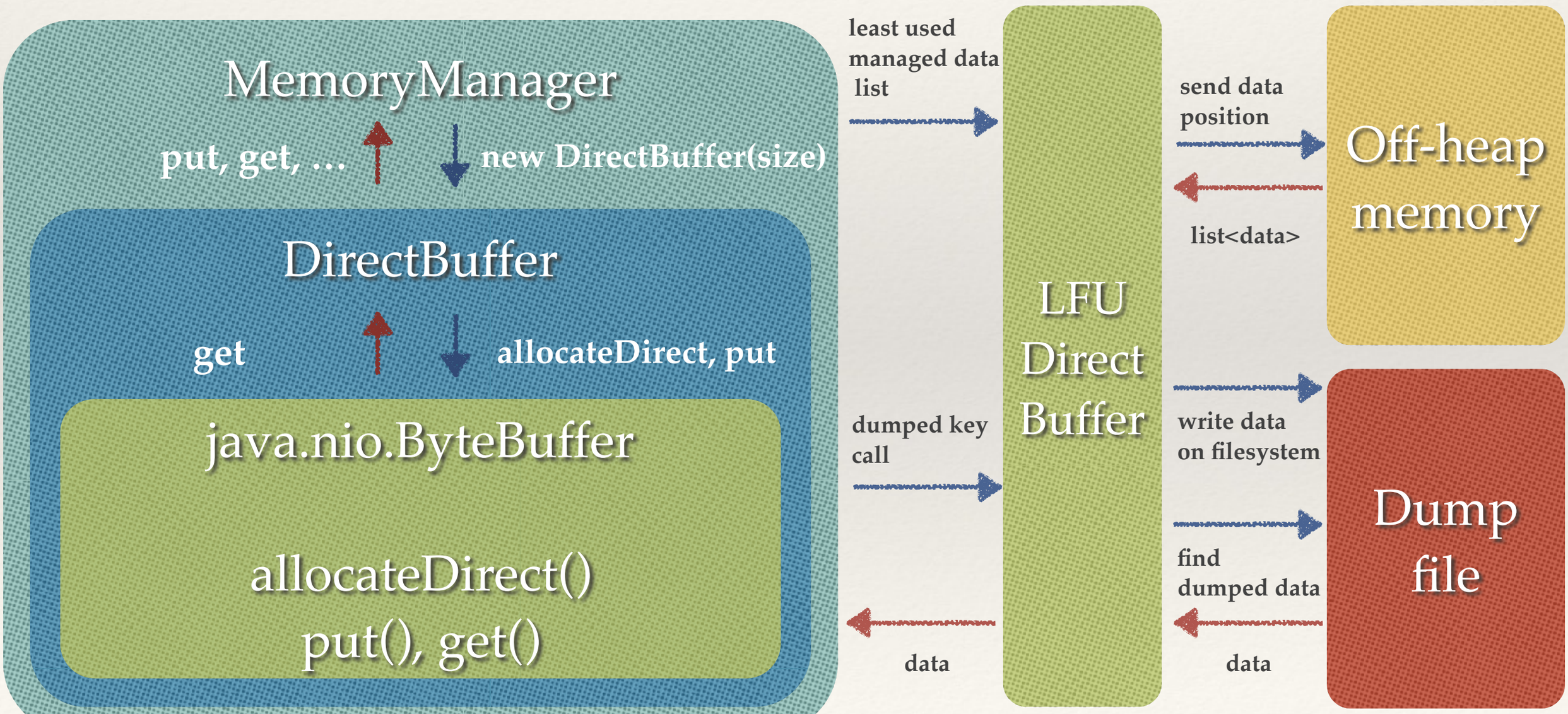
DirectBuffer의 단편화를 해결하기 위한 **PullDirectBuffer** 클래스를 개발



개발 내용

- Off-heap 메모리를 구현하기 위해 direct buffer를 사용한 Memory Allocator 개발

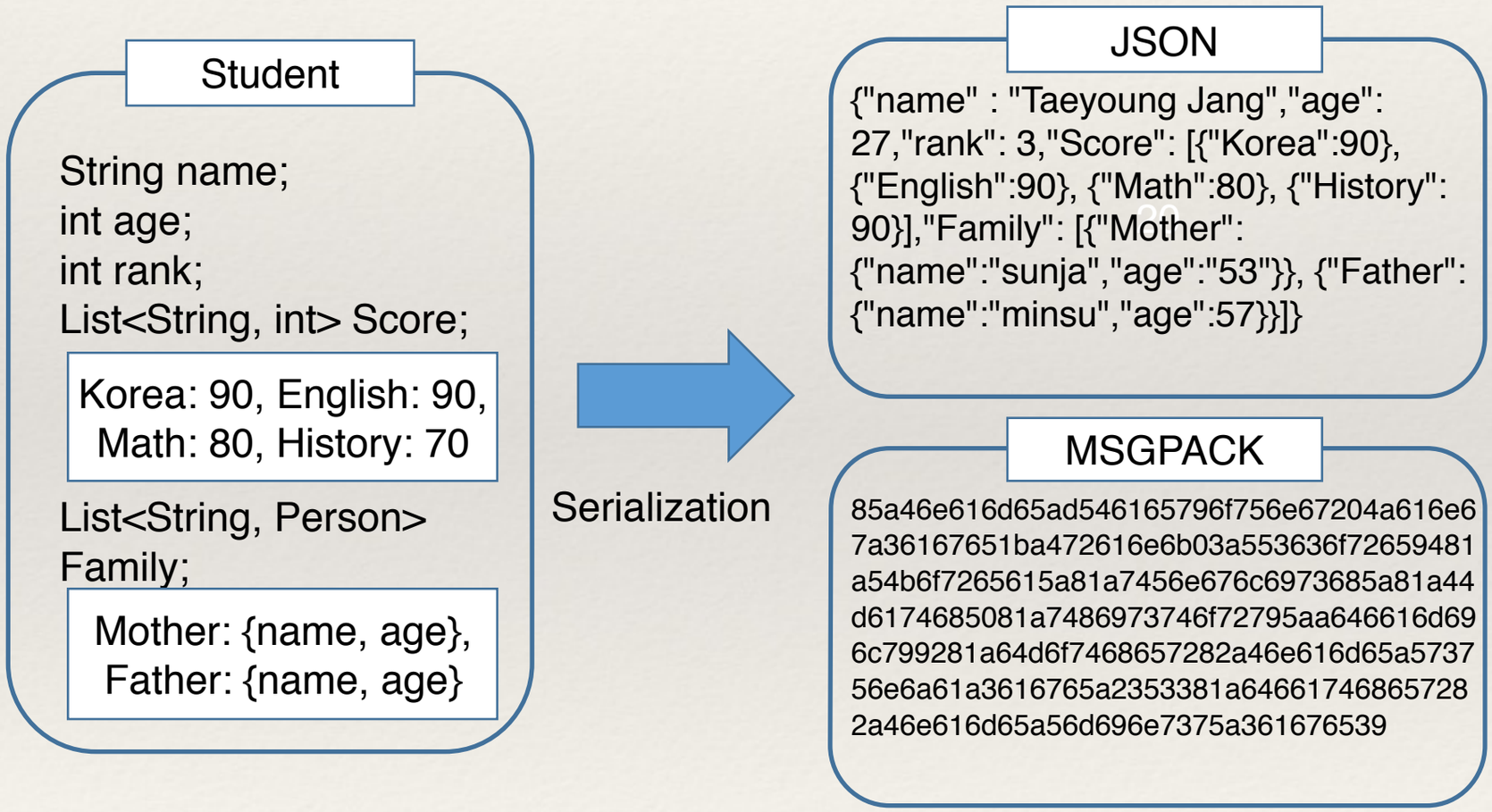
용량 부족으로 인한 데이터 소실 및 저장 불가를 위한 **LFU(Least Frequently Used)** 기능 구현



개발 내용

- Data Structure를 기반으로 한 인터페이스 제공

  기존 IMDG 의 데이터 저장 방식: Serialization

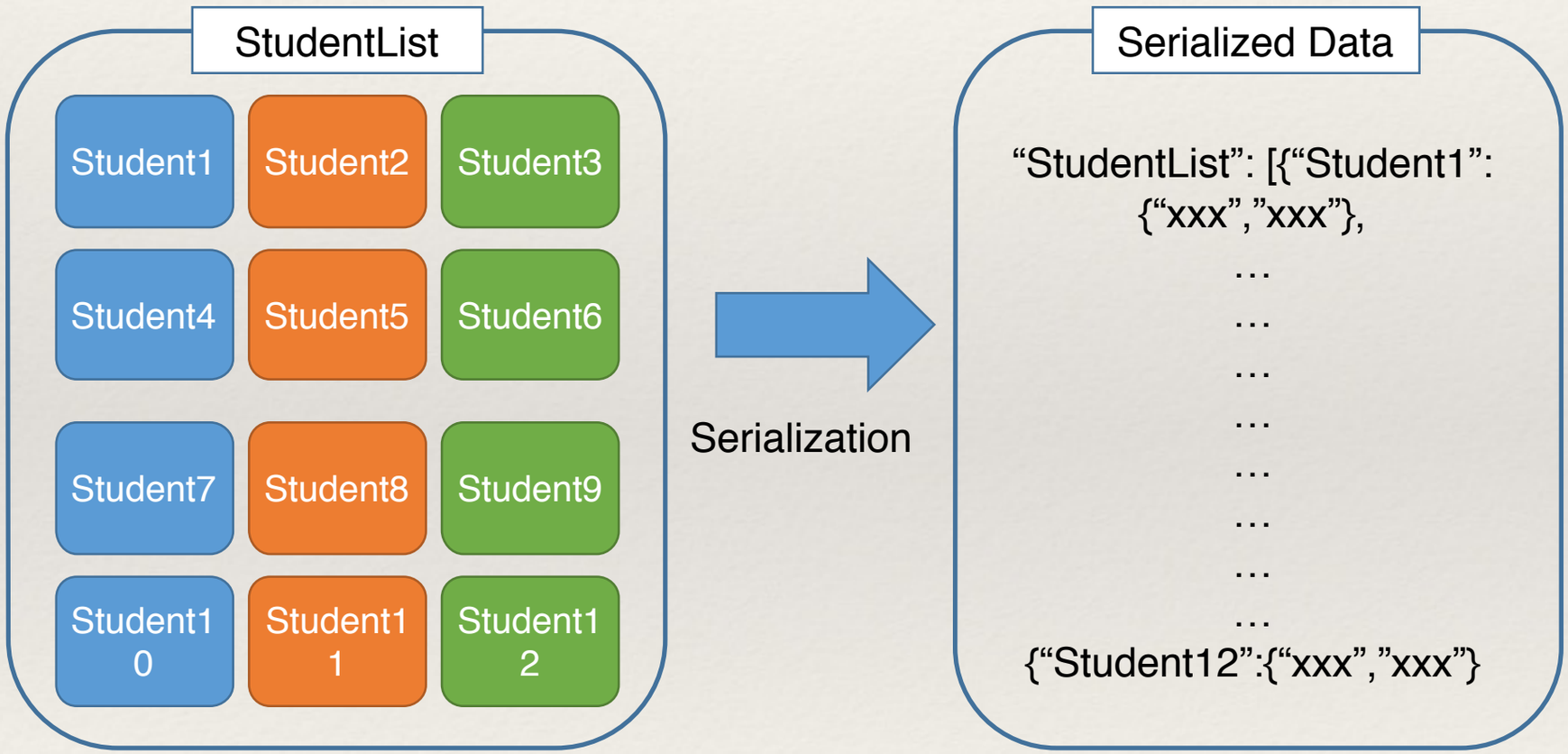


객체의 Serialization을 통해서 DB에 Key-Value pair로 저장을 한다.

개발 내용

- Data Structure를 기반으로 한 인터페이스 제공

  기존 IMDG 의 데이터 저장 방식: Serialization

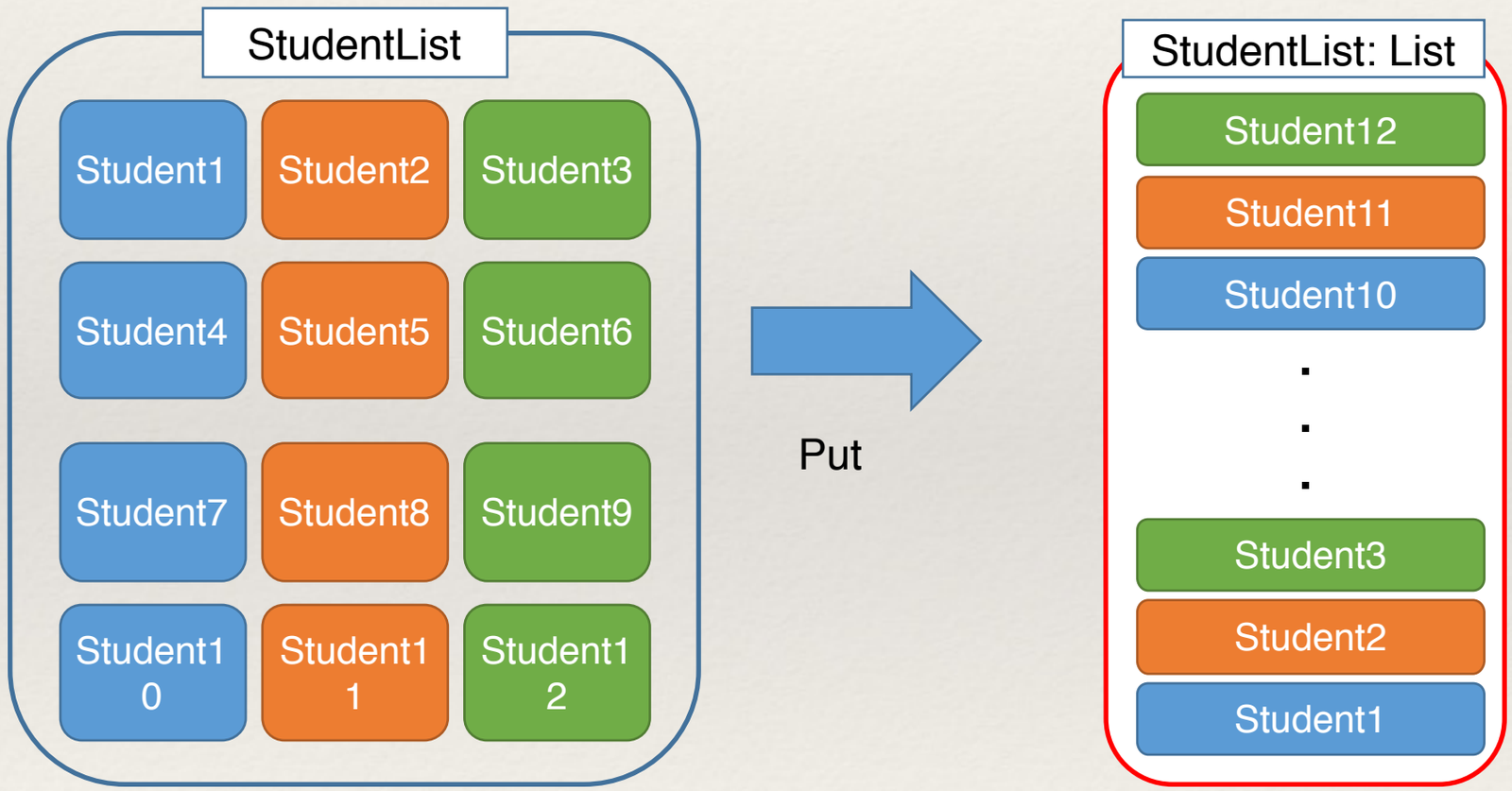


여기서 3번 학생의 정보만 얻어오기 위해선 모든 객체를 다 받아온 후,
Deserialization 후 얻어와야 하기 때문에 패킷 낭비나 클라이언트의 부담이 크다

개발 내용

- Data Structure를 기반으로 한 인터페이스 제공

  Greenland IMDG의 데이터 저장 방식: 자료구조 이용



해당 자료구조에 전송할 때는 Serialization해서 전송 후, 서버에서 자료구조를 구축 후 클라이언트에서 얻어올 때는 자료구조의 함수를 이용하듯 얻어와 구조를 개선한다

개발 내용

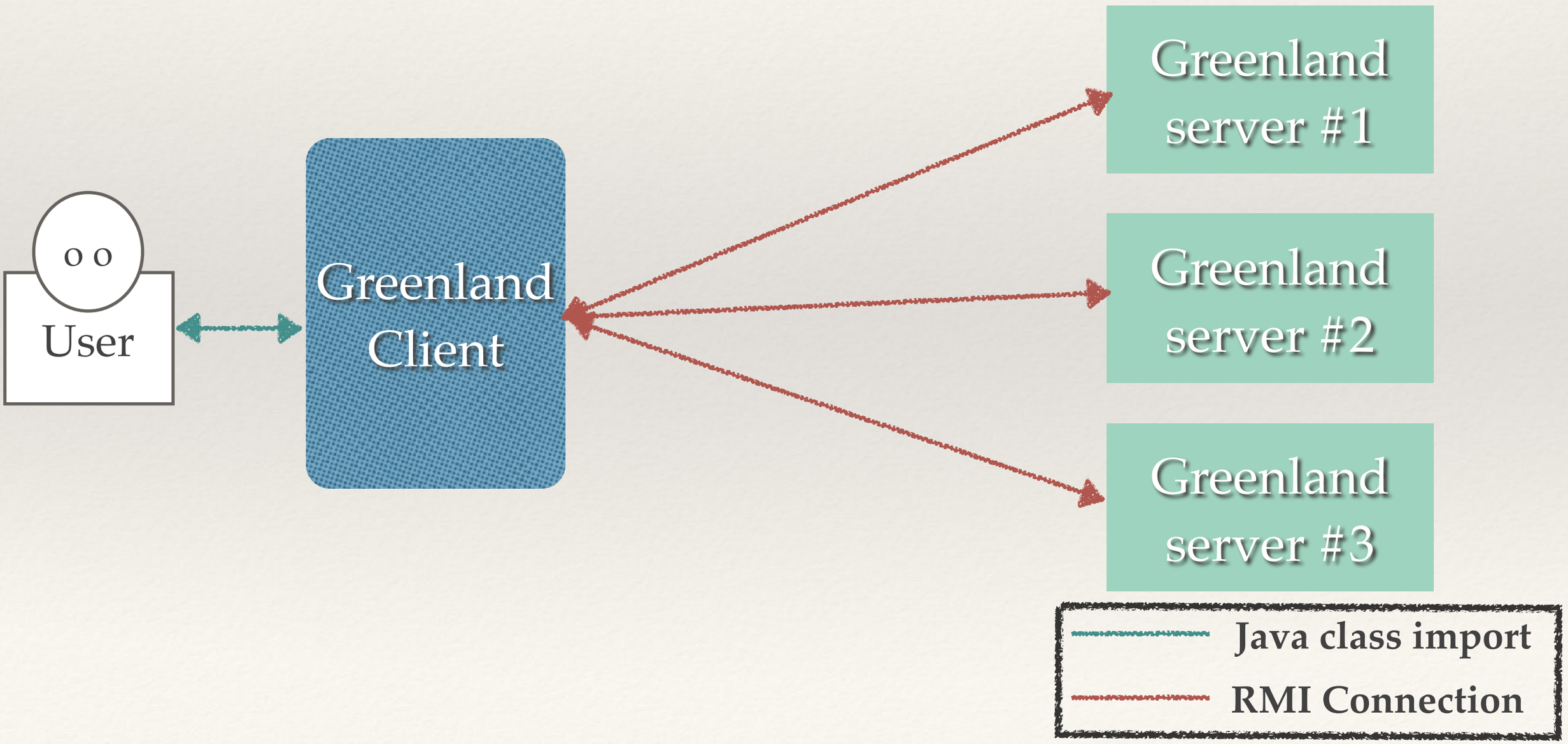
- Data Structure를 기반으로 한 인터페이스 제공

CommandBuilder interface

Keys	Hashes	Sorted Set
SET	HGET	ZADD
GET	HSET	ZREM
EXIST	HEXISTS	ZRANK
DEL	HLEN	ZRANGE
EXPIRE	HDEL	ZCARD
.	.	.
.	.	.

개발 내용

- Java에서 IMDG에 접근하고 이용할 수 있는 라이브러리와 Client 개발



개발 내용

- IMDG의 모니터링과 관리를 위한 웹 콘솔 개발

jQueryUI

jQuery ComboBox

Command Builder

jQuery

Bootstrap

maani.us

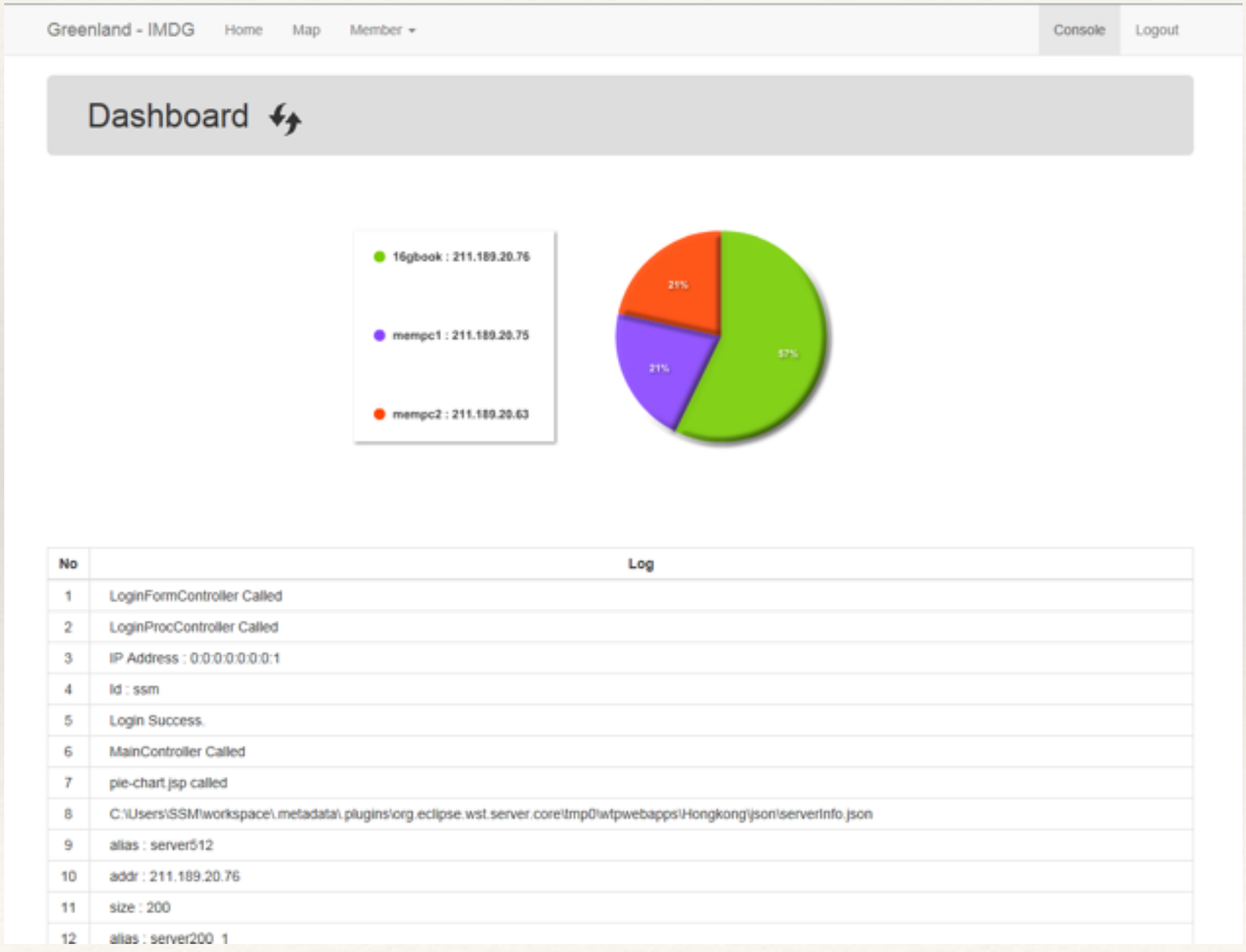
Instance Connection List

Spring frameworks 3.1

Instance Manager

개발 내용

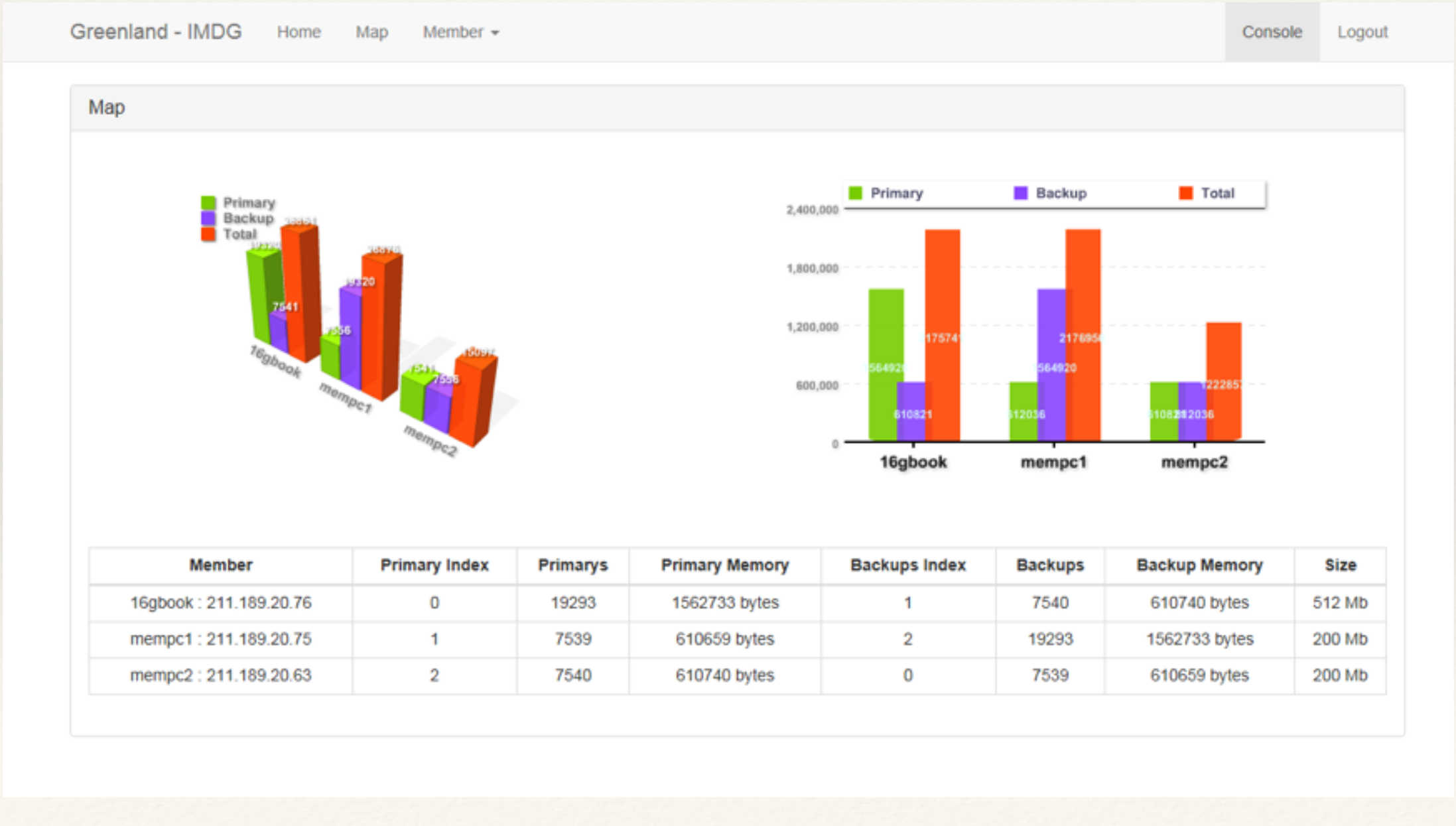
- IMDG의 모니터링과 관리를 위한 웹 콘솔 개발



No	Log
1	LoginFormController Called
2	LoginProcController Called
3	IP Address : 0:0:0:0:0:0:0:1
4	id : ssm
5	Login Success.
6	MainController Called
7	pie-chart.jsp called
8	C:\Users\SSM\workspace\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps\Hongkong\json\serverInfo.json
9	alias : server512
10	addr : 211.189.20.76
11	size : 200
12	alias : server200_1

개발 내용

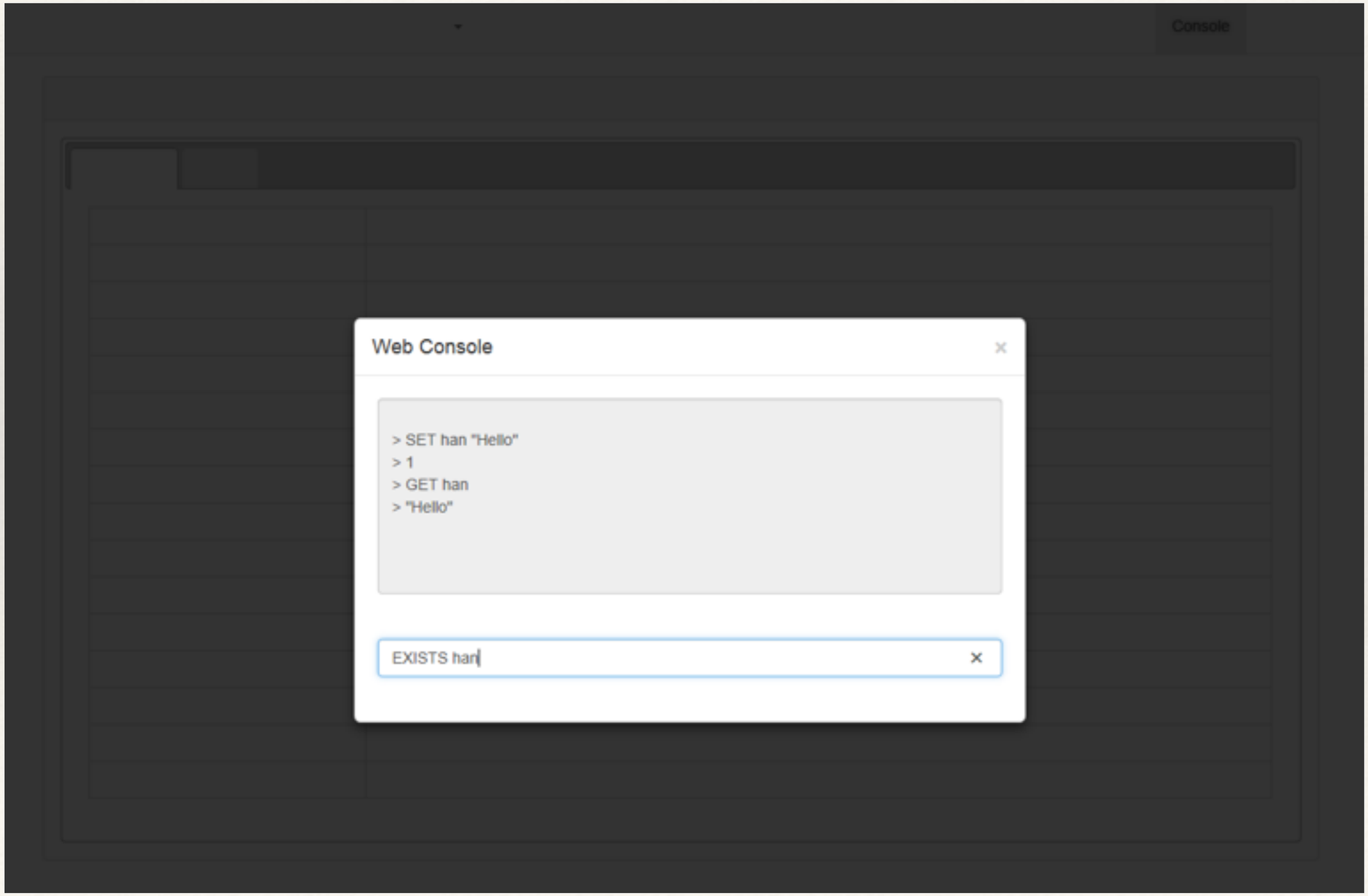
- IMDG의 모니터링과 관리를 위한 웹 콘솔 개발



Member	Primary Index	Primarys	Primary Memory	Backups Index	Backups	Backup Memory	Size
16gbook : 211.189.20.76	0	19293	1562733 bytes	1	7540	610740 bytes	512 Mb
mempc1 : 211.189.20.75	1	7539	610659 bytes	2	19293	1562733 bytes	200 Mb
mempc2 : 211.189.20.63	2	7540	610740 bytes	0	7539	610659 bytes	200 Mb

개발 내용

- IMDG의 모니터링과 관리를 위한 웹 콘솔 개발

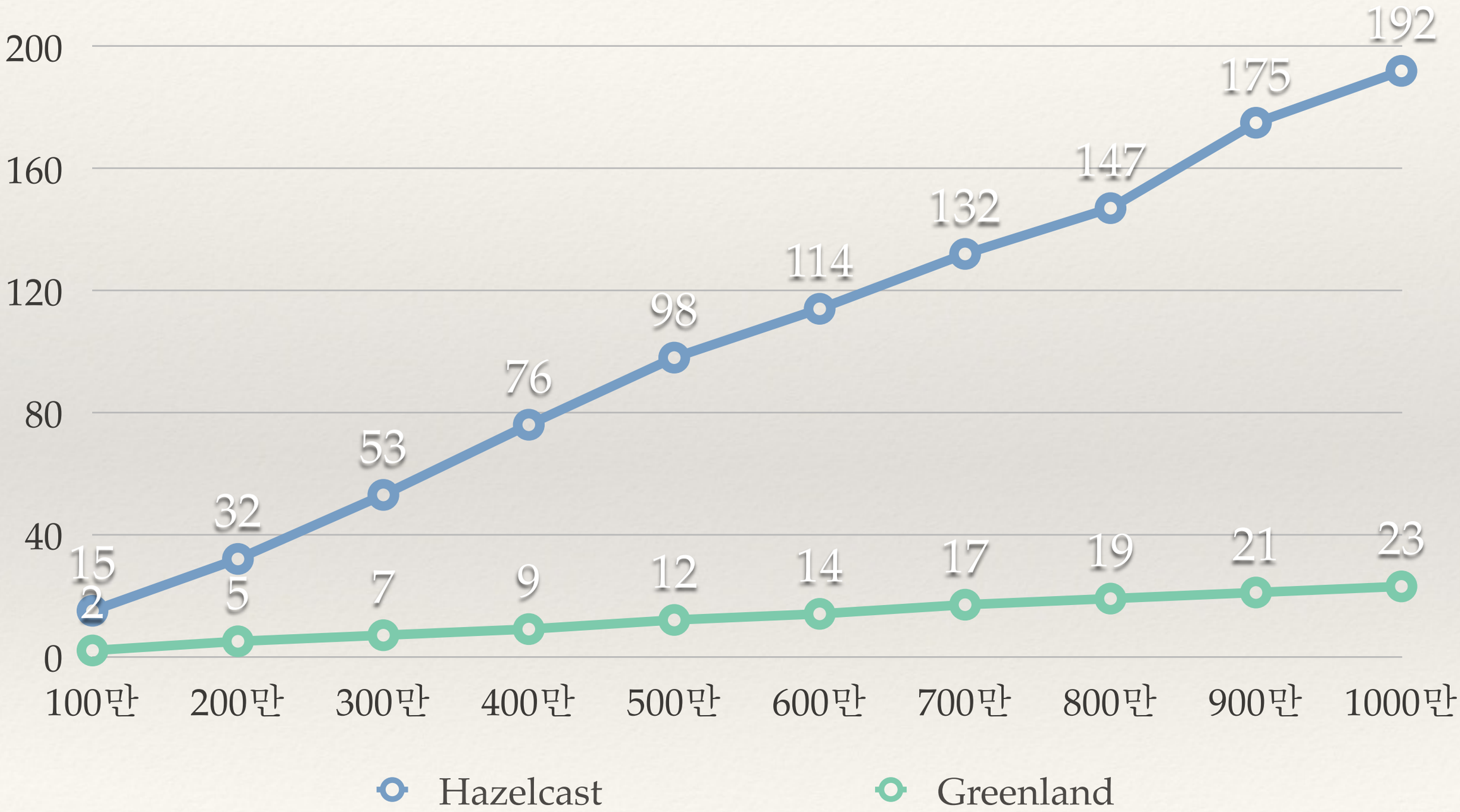


벤치마킹

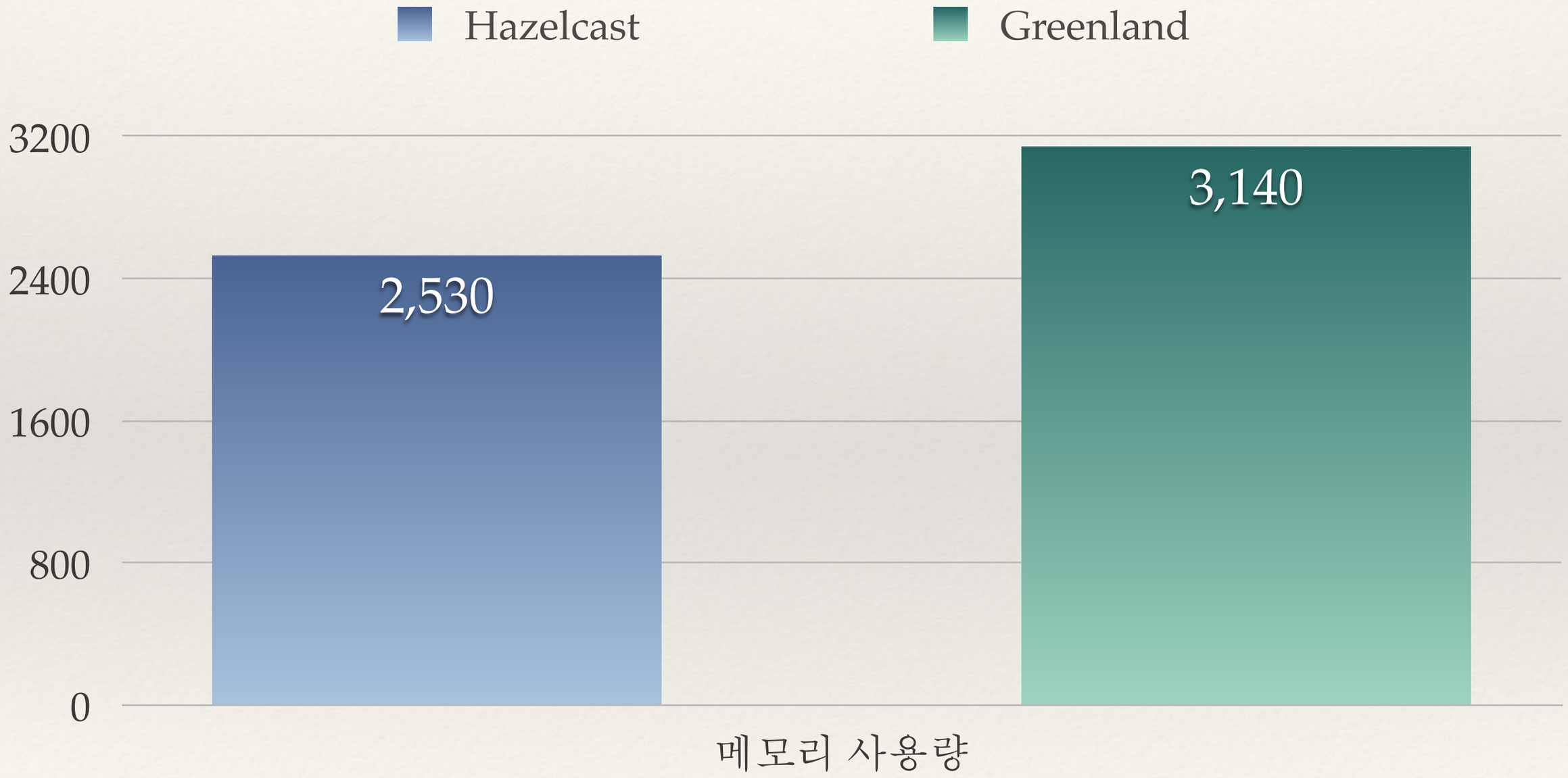
- Hazelcast와 1000만 개 삽입 속도를 로컬에서 비교
- 100만개 마다 시간을 재서 각각의 최대/최소값 확보
- 메모리 사용량 표기

	총 소요 시간	최소 시간	최대 시간	메모리 사용량
Greenland	20초	2초	3초	3.14GB
Hazelcast	175초	15초	23초	2.53GB

벤치마킹


200
160
120
80
40
0
100만
200만
300만
400만
500만
600만
700만
800만
900만
1000만
15
32
53
76
98
114
132
147
175
192
2
5
7
9
12
14
17
19
21
23
Hazelcast
Greenland

벤치마킹



감사합니다

Q&A