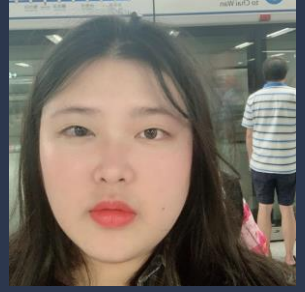
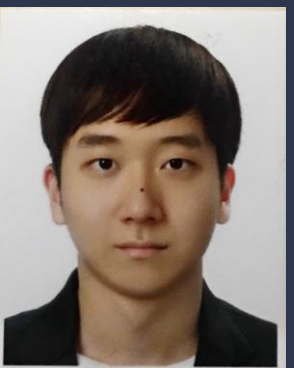


최종발표

KUORA

9 팀 (서지원, 홍나리) ㅣ 졸업프로젝트 1 ㅣ 유준범 교수님

2020.06.18



Contents

Concept

KUORA

Google File System

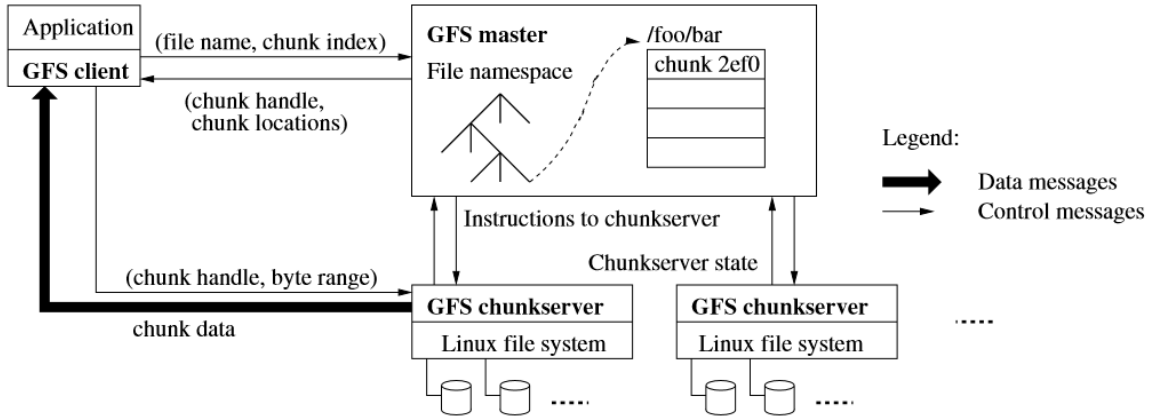
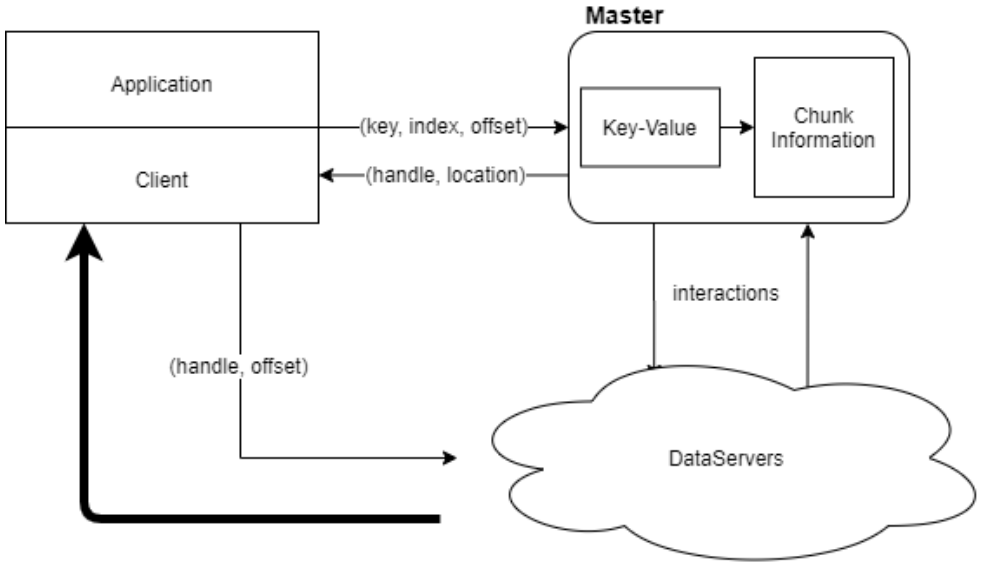


Figure 1: GFS Architecture

KUORA



Assumption

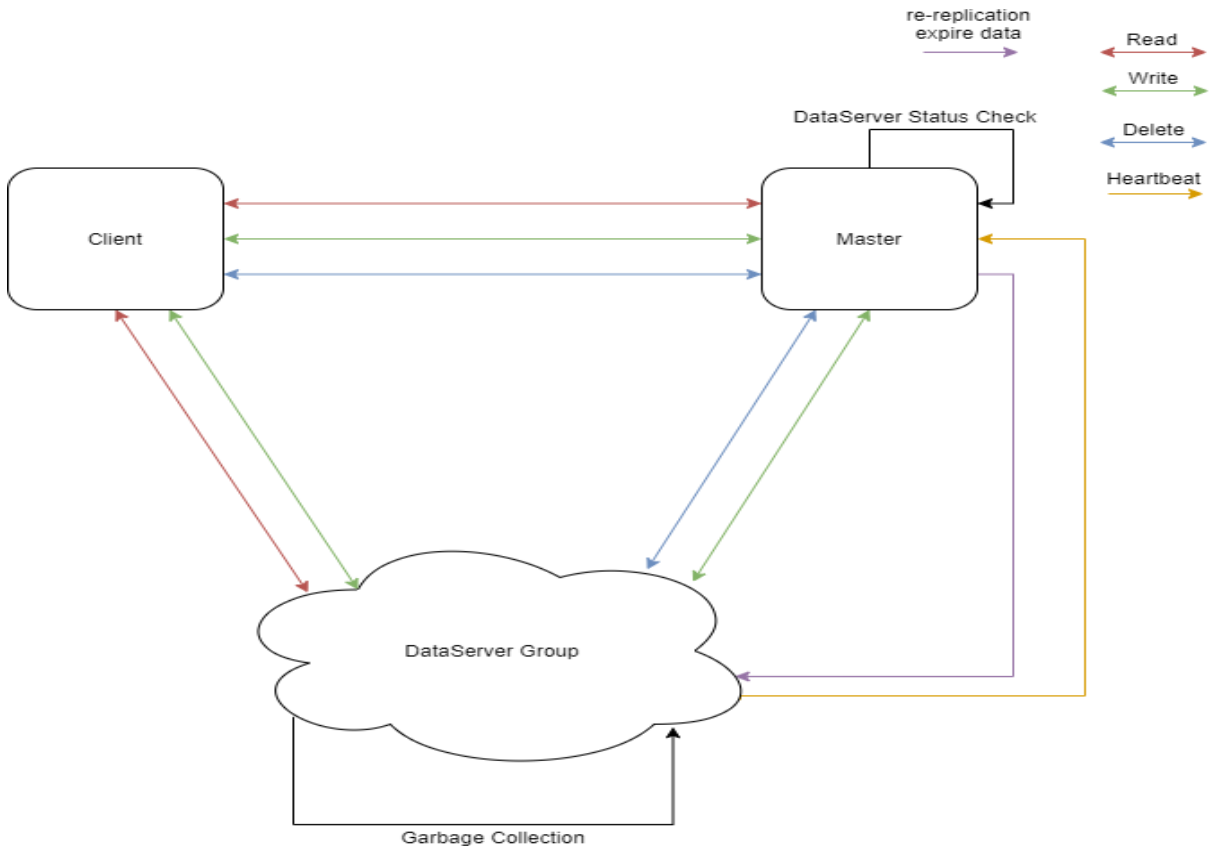
- 단, 파일(이미지)의 수정은 없는 것으로 간주
- Metadata : Key-value 방식

Another Distributed

KUORA

Goal

KUORA



Deliverable

저장소 가용률을 위한 DFS의 구현

- Read File, Write File, Delete File 구현
- Replication, Re-Replication 구현

Requirements

KUORA

• Functional Requirements

1. Client Side

1.1. Read File
- Client가 File Key를 통하여 System으로부터 파일의 데이터를 읽는다.

1.2. Write File
- Client가 System에 새로운 File을 만들고 해당 File에 데이터를 쓴다.

1.3. Delete File
- Client가 File Key를 통하여 System으로부터 파일을 삭제한다.

2. System Side (Master + DataServer)

2.1. Heartbeat
- DataServer는 Master에게 자신의 정상 동작 여부를 전송한다. (400 ~ 800ms)

2.2. Expire/Migration
- System은 Hot File의 경우 일정 기간이 지난 후 Cold Storage로 이동한다.

2.3. Garbage Collection
- DataServer는 주기적으로 삭제/만료된 파일에 대하여 가비지 콜렉션을 시행한다.

2.4. Replication
- 처음 파일이 생성되고 데이터가 쓰여질 때, 데이터의 복사본을 생성하여 다른 DataServer에 저장한다.

2.5. Re-Replication
- DataServer에 장애가 발생할 시, Master는 장애가 발생한 DataServer가 가진 데이터에 대하여 정상인 DataServer가 복제를 수행하도록 지시한다.

• Quality Requirements

Q1. Scalability

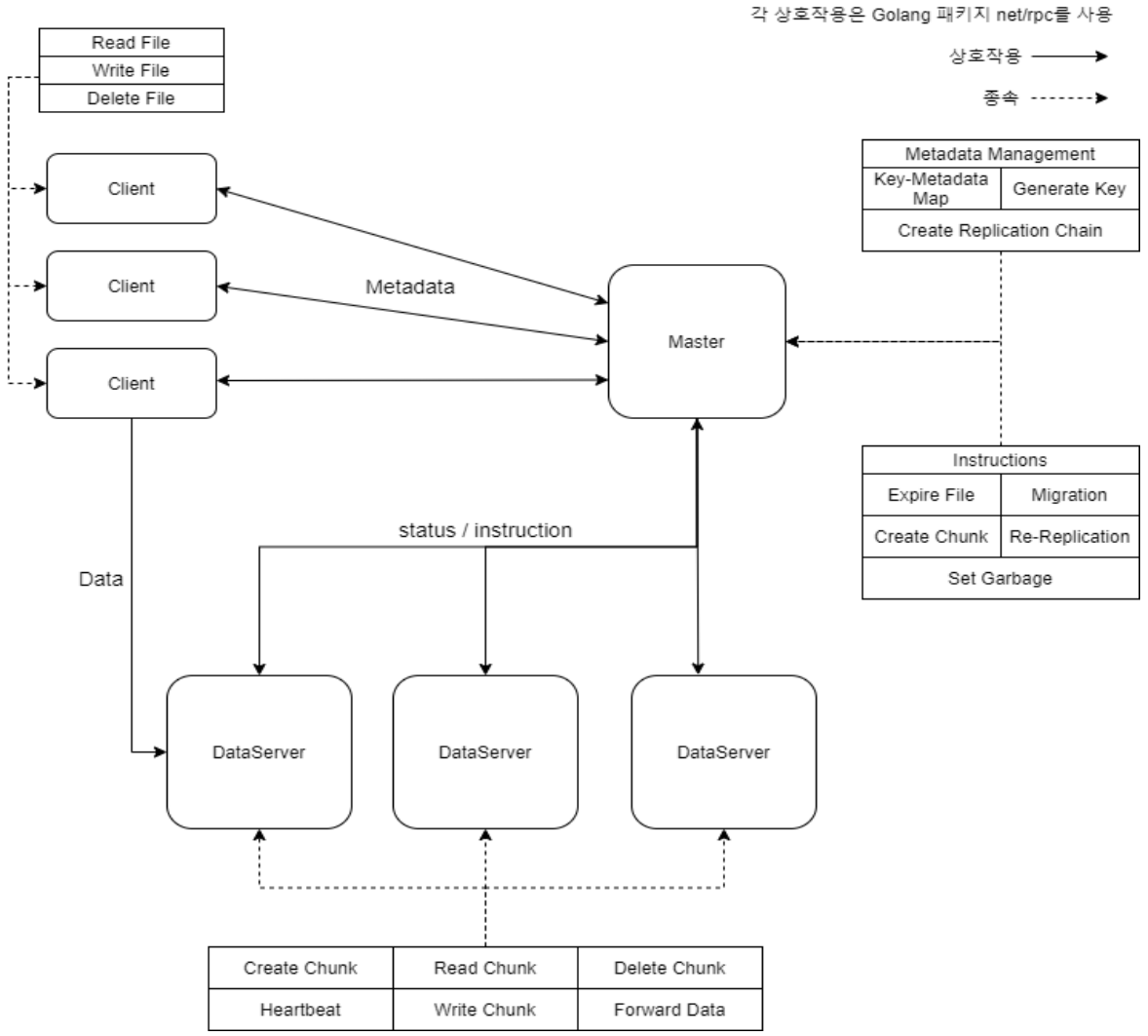
- Master는 새로운 DataServer가 실행될 경우 이를 시스템에 포함시킬 수 있다.

Q2. Atomicity

- 다수의 Client가 파일 스토리지에 접근할 수 있다.
- Master와 DataServer는 Chunk의 정보에 대하여 R/W Lock을 가진다.
- Lock을 통해서 각 Chunk에 대해서 Client의 연산에 대해서 Atomicity를
보장할 수 있도록 한다.

Architecture Diagram

KUORA



Client

- Read/Write/Delete File
- Client-Master 간의 Metadata 교환 후 Client-DataServer 간 직접 통신

Master

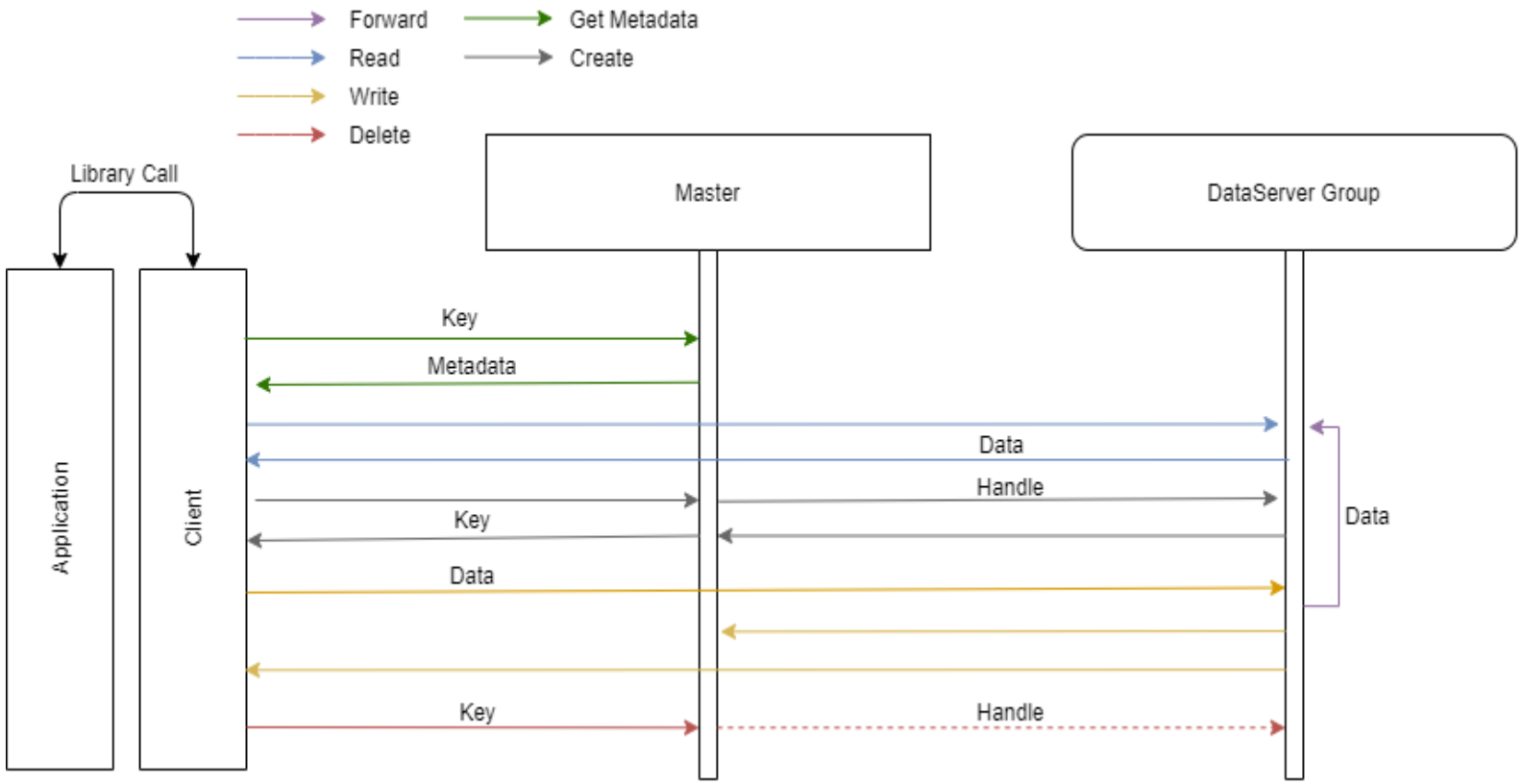
- File Key와 Metadata간 Mapping 관리
- DataServer의 정상 동작을 관리
- DataServer의 작업에 대한 지시

DataServer

- 실질적인 파일의 데이터를 Chunk로 관리
- 각 데이터에 대한 Create/Read/Write/Delete
- 다른 DataServer로의 데이터의 전달

System Sequence Diagram

KUORA



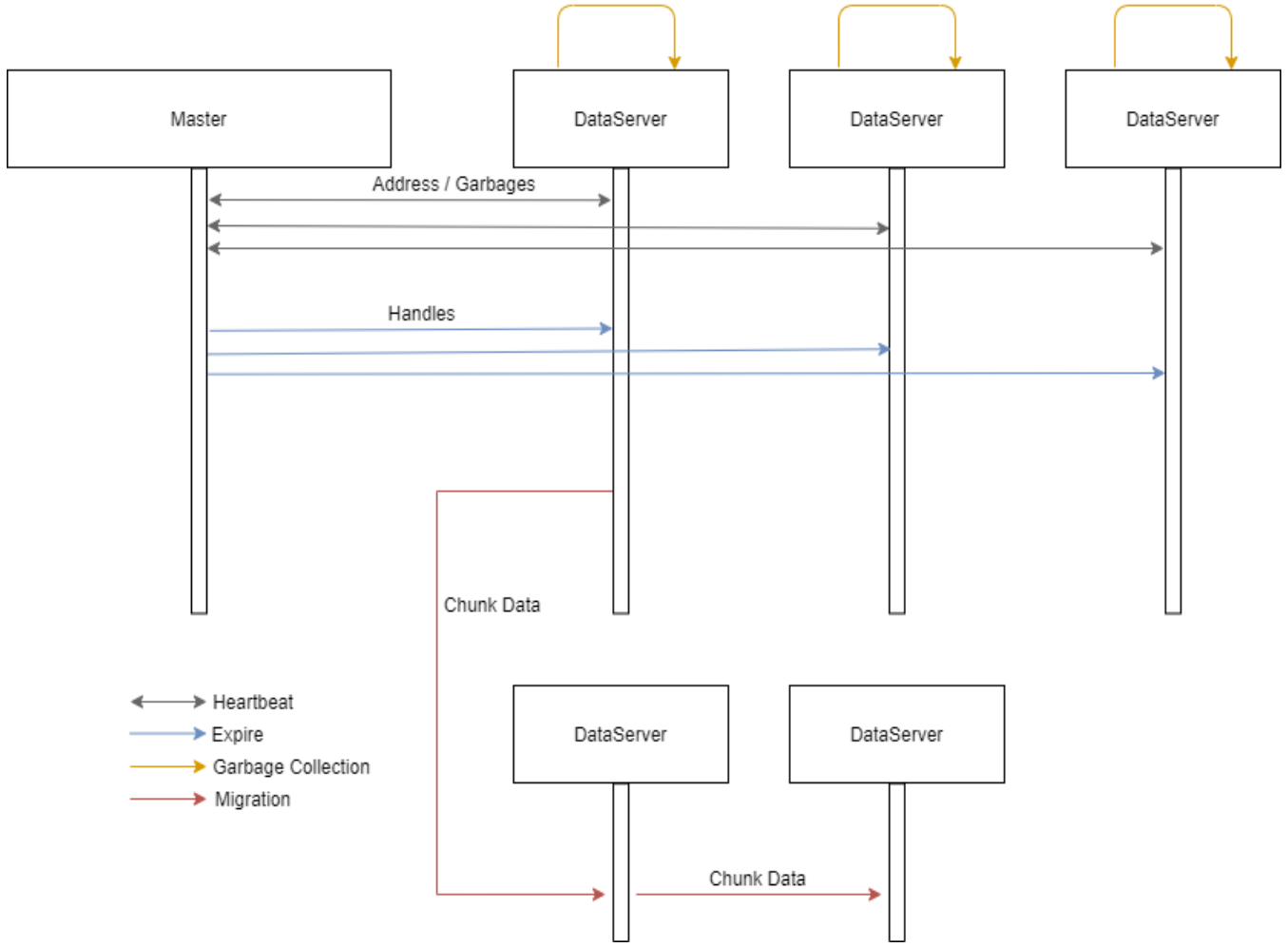
Read
1. Exchange Metadata
2. Read Data

Write
1. Create File
2. Write File
3. Replication

Delete
1. Delete Request

System Sequence Diagram

KUORA



Heartbeat

1. DataServer가 Master에게 Address 및 상태 전송
2. Master는 DataServer에 Garbage (Chunks) 전송

Expire/Migration

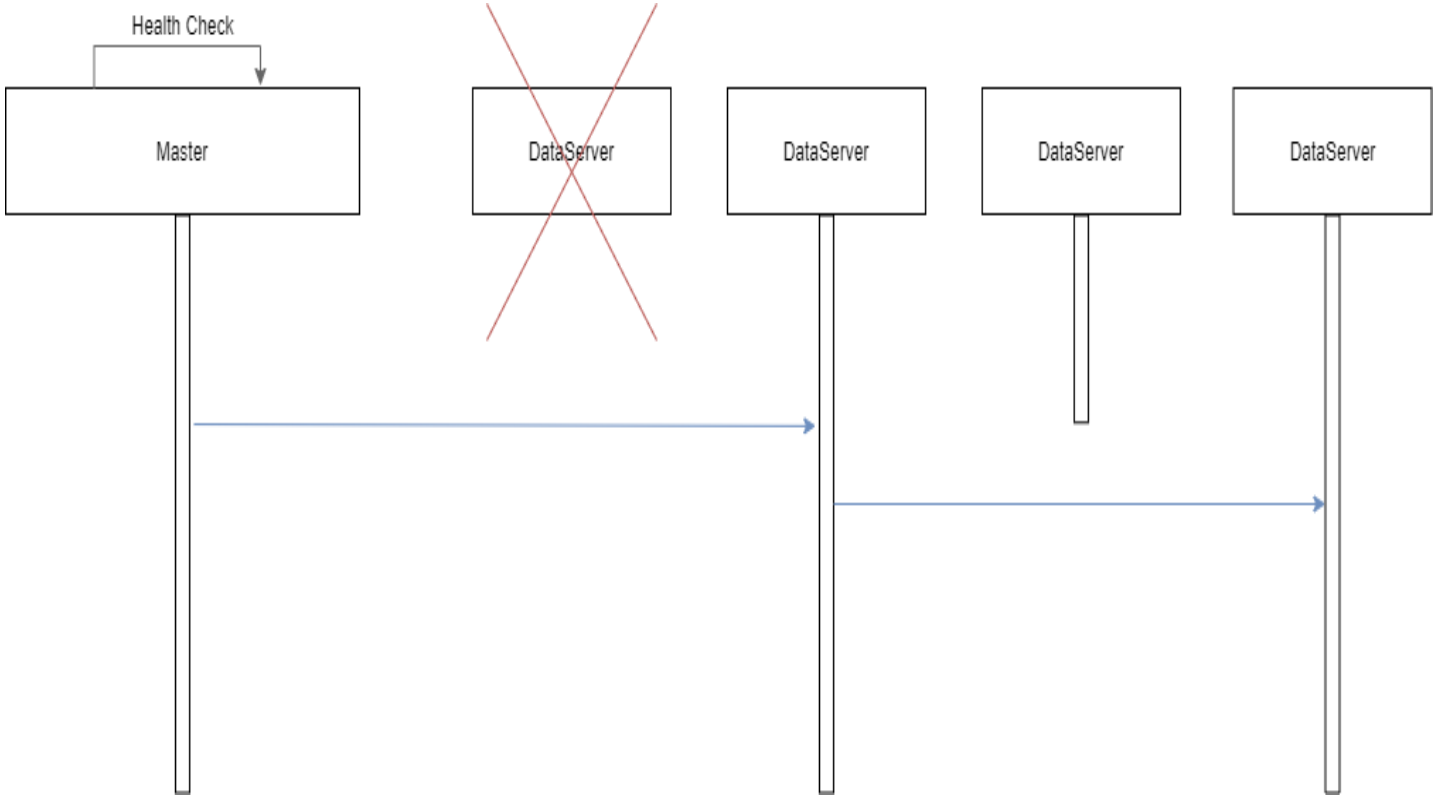
1. Master는 주기적으로 Chunk의 Expire 여부 체크
2. Expired Chunks를 Migration하도록 DataServer에 지시
3. DataServer가 Migration 수행

Garbage Collection

1. DataServer가 Master에게 전송받은 Garbage chunks에 대해 주기적으로 Garbage Collection 수행

System Sequence Diagram

KUORA



IF.

DataServer 중 하나가 비정상적으로 Shutdown 되었을 시

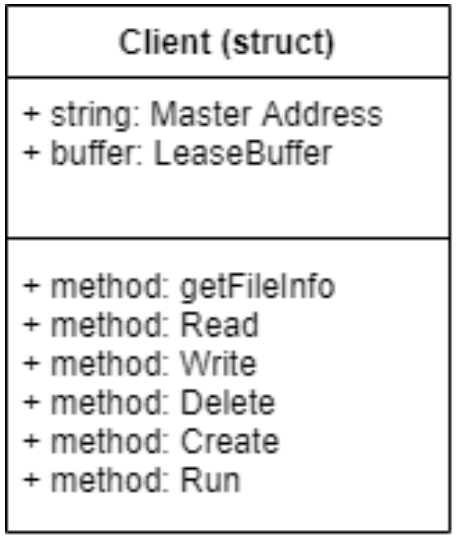
1. Heartbeat의 전송이 끊기며 Master가 Health Check를 통해 장애(Fault) 파악

2. Fault DataServer가 가진 Chunk에 대해서 이를 가지고 있는 다른 DataServer에게 복제를 지시

3. DataServer가 이를 수행

Low-Level Design

KUORA



Client

1.1 getFileInfo
Key를 통해 File의 Metadata 얻을 수 있음.

1.2 Read
Chunk를 Read

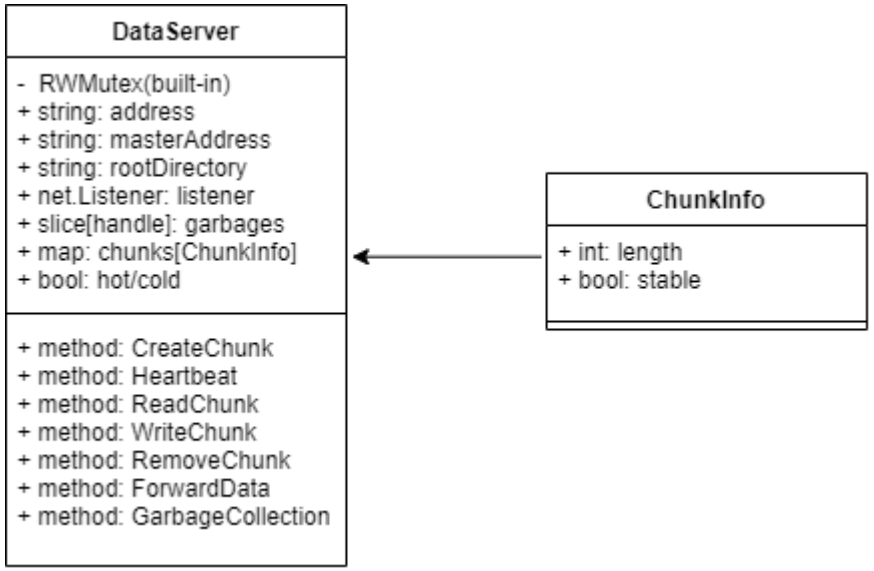
1.3 Write
Chunk에 data를 Write

1.4 Delete
Key를 통해 File을 지우도록 요청.

1.5 Create
Master에 File을 생성하도록 요청.

Low-Level Design

KUORA



DataServer

2.1 CreateChunk
빈 Chunk를 생성.

2.2 Heartbeat
Master와 주기적으로 정보 교환.

2.3 ReadChunk
Chunk의 데이터를 읽어서 Client에 전송.

2.4 WriteChunk
Chunk에 데이터를 씀.

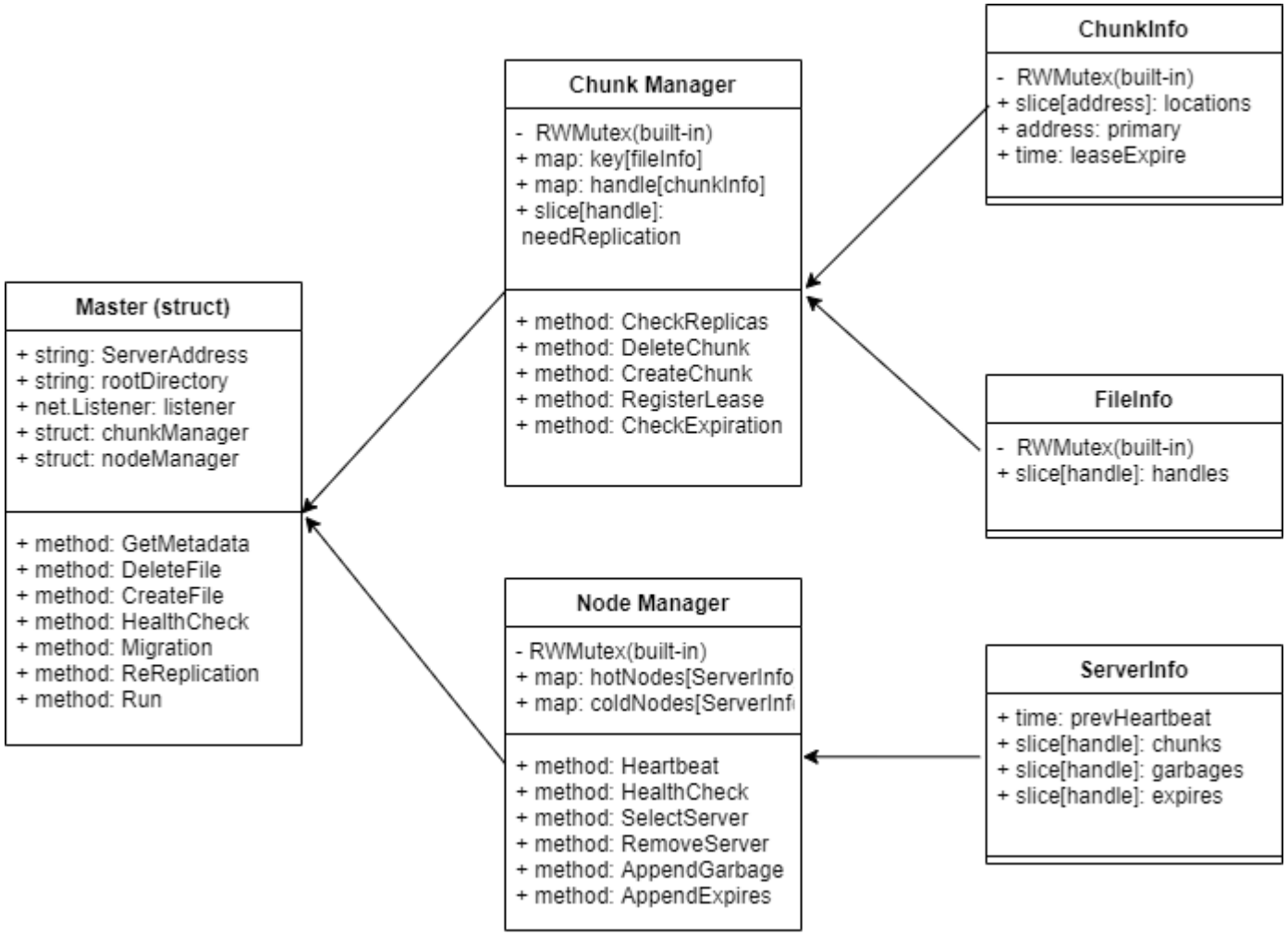
2.5 RemoveChunk
Chunk를 삭제

2.6 ForwardData
다른 DataServer로 Chunk data 전송.

2.7 Garbage Collection
주기적으로 실행

Low-Level Design

KUORA



Master

3.1 GetMetadata
Client의 요청에 따라 Metadata 제공

3.2 DeleteFile
Key와 함께 논리적인 File을 삭제, DataServer에 Garbage 전송

3.3 CreateFile
Key와 함께 논리적인 File을 생성

3.4 HealthCheck
Heartbeat를 통하여 각 서버의 상태 체크

3.5 Migration
Expired Chunks를 파악하여 Migration을 지시

3.6 ReReplication
DataServer에 ReReplication을 지시

Test Cases

KUORA

No.	Name	Description
1.1	Read File	Client에서 Key를 통해 Read를 요청하였을 때 해당 데이터를 읽어오는가 여부를 확인
1.2	Write File	Client에서 새로운 파일을 만들고 Write 요청을 할 때 해당 데이터가 DataServer에 올바르게 저장되는가 여부를 확인
1.3	Delete File	Client에서 Key를 통해 Delete 요청을 처리 후 해당 데이터가 Garbage Collection 되는지 확인
2.1	Heartbeat	Master가 지속적으로 DataServer로부터 요청을 받아 DataServer가 정상임을 확인할 수 있는지 여부를 확인
2.2	Expire/Migration	Master의 지시에 따라 DataServer가 Expire에 의한 Delete 혹은 Migration을 수행할 수 있는지 여부를 확인
2.3	Garbage Collection	Configure된 주기에 따라 DataServer 내의 실제 Chunk 데이터가 삭제되는지 여부를 확인
2.4	Replication	Client의 Write 후 DataServer에 Chunk 데이터가 복제가 되었는지 확인
2.5	Re-Replication	DataServer 하나를 중지시킨 후 해당 DataServer가 가진 데이터가 다른 서버에 복제되었는지 확인
Q1	Persistent Metadata Test	Disk 내의 파일을 Serialize 하여 Metadata 정보가 동일한지 확인
Q2	Multiple Client Test	Client를 다수 실행하여 쓰기 및 읽기를 실행하였을 때 데이터의 동일성 및 Key의 유일성을 확인

Test Flow
: Master와 DataServer는 각 연산에 대해 Logging을 수행한다.

1. Client Test
   - Write File / Replication

   Client의 요청에 따라 논리적인 File에 대한 Key를 발급하고 더미 데이터의 write와 복제되는 동작을 log를 통해 확인한다.
   - Read File

   Client가 Write하고 받은 key를 바탕으로 Read를 수행한다. 이 때 데이터가 쓰여진 3개의 서버 모두에서 read를 수행하여 Client가 쓴 파일의 데이터와 일치하는지 확인한다.
   - Delete File

   Client가 Key를 통해 파일을 삭제하고 key를 Listing하여 삭제 여부를 확인한다. 또한, Garbage Collection 이후 해당 Chunk가 실제로 삭제되었는지 확인한다.
   - 여러 Thread를 생성하여 Client의 기능을 수행하여 Multiple Client 테스트를 수행한다.

2. System Test
   - Heartbeat

   Master의 Log를 통해 확인
   - Expire/Migartion/Garbage Collection

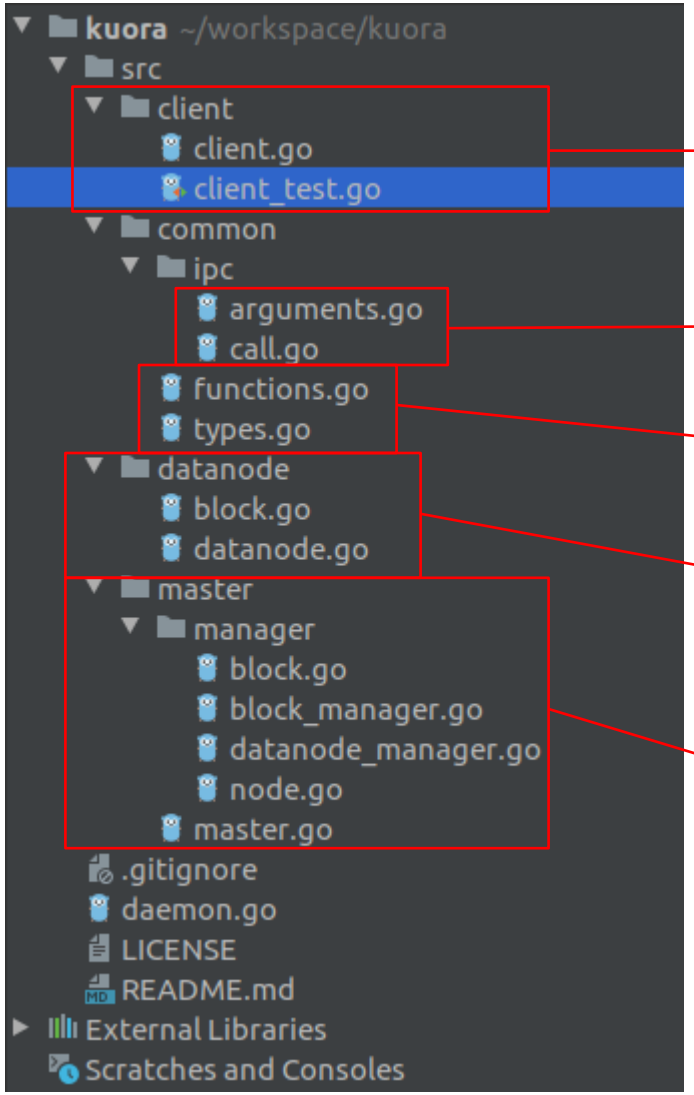
   Master와 DataServer의 Log와 실제 DataServer에 접속하여 Linux 내 파일의 여부를 확인
   - Re-Replication

   DataServer 중 하나를 강제종료한 후 가지고 있던 Chunk들이 복제되는지 확인
   - Persistent Metadata

   Master/DataServer 종료 후 정상적으로 disk의 metadata를 메모리에 저장할 수 있는지 확인

Packages

KUORA



Client Package
- Client 기능 / Client Test Code

IPC Package
- RPC Call 기능 / RPC Data Structure

common Package
- User Defined Type / 공용 함수

datanode Package
- DataNode의 기능 및 Data Structure

master Package
- Master 기능 및 Data Structure

Test Log

KUORA

```
INFO[0000] Run Master Node - - -
INFO[0000] INIT NEW BLOCK MANAGER
INFO[0000] INIT NEW DATANODE MANAGER
2020/06/04 00:14:41 rpc.Register: method "ExpireCheck" has 1 input parameters; needs exactly three
INFO[0002] New DataNode 127.0.0.1:40002
INFO[0005] New DataNode 127.0.0.1:40001
INFO[0019] Create File Operation
INFO[0019] map[127.0.0.1:40001:{0} 127.0.0.1:40002:{0}]
INFO[0019] map[]
INFO[0019] Create Block Info - &{{{0 0} 0 0 0 0} 2020-06-04 00:18:00.724629909 +0900 KST m=+199.117396830 [127.0.0.1:40001 127.0.0.1:40002] 127.0.0.1:40001}
INFO[0019] Hot Key : 0_0
INFO[0019] Block Information - &{{{0 0} 0 0 0 0} 2020-06-04 00:18:00.724629909 +0900 KST m=+199.117396830 [127.0.0.1:40001 127.0.0.1:40002] 127.0.0.1:40001}
INFO[0019] Hot Key : 0_0
INFO[0019] Block Information - &{{{0 0} 0 0 0 0} 2020-06-04 00:18:00.724629909 +0900 KST m=+199.117396830 [127.0.0.1:40001 127.0.0.1:40002] 127.0.0.1:40001}
INFO[0019] Create File Operation
INFO[0019] map[127.0.0.1:40001:{0} 127.0.0.1:40002:{0}]
INFO[0019] map[]
INFO[0019] Create Block Info - &{{{0 0} 0 0 0 0} 2020-06-04 00:18:00.759382461 +0900 KST m=+199.152149380 [127.0.0.1:40001 127.0.0.1:40002] 127.0.0.1:40001}
INFO[0019] Create Block Info - &{{{0 0} 0 0 0 0} 2020-06-04 00:18:00.759413993 +0900 KST m=+199.152180913 [127.0.0.1:40001 127.0.0.1:40002] 127.0.0.1:40001}
INFO[0019] Create Block Info - &{{{0 0} 0 0 0 0} 2020-06-04 00:18:00.75942652 +0900 KST m=+199.152193441 [127.0.0.1:40001 127.0.0.1:40002] 127.0.0.1:40001}
INFO[0019] Hot Key : 1_2_3_0
INFO[0019] Block Information - &{{{0 0} 0 0 0 0} 2020-06-04 00:18:00.759382461 +0900 KST m=+199.152149380 [127.0.0.1:40001 127.0.0.1:40002] 127.0.0.1:40001}
INFO[0019] Block Information - &{{{0 0} 0 0 0 0} 2020-06-04 00:18:00.759413993 +0900 KST m=+199.152180913 [127.0.0.1:40001 127.0.0.1:40002] 127.0.0.1:40001}
INFO[0019] Block Information - &{{{0 0} 0 0 0 0} 2020-06-04 00:18:00.75942652 +0900 KST m=+199.152193441 [127.0.0.1:40001 127.0.0.1:40002] 127.0.0.1:40001}
INFO[0019] Hot Key : 1_2_3_0
INFO[0019] Block Information - &{{{0 0} 0 0 0 0} 2020-06-04 00:18:00.759382461 +0900 KST m=+199.152149380 [127.0.0.1:40001 127.0.0.1:40002] 127.0.0.1:40001}
INFO[0019] Hot Key : 0_0
INFO[0019] Block Information - &{{{0 0} 0 0 0 0} 2020-06-04 00:18:00.724629909 +0900 KST m=+199.117396830 [127.0.0.1:40001 127.0.0.1:40002] 127.0.0.1:40001}
INFO[0019] Hot Key : 1_2_3_0
INFO[0019] Block Information - &{{{0 0} 0 0 0 0} 2020-06-04 00:18:00.759382461 +0900 KST m=+199.152149380 [127.0.0.1:40001 127.0.0.1:40002] 127.0.0.1:40001}
INFO[0019] Hot Key : 0_0
INFO[0019] Block Information - &{{{0 0} 0 0 0 0} 2020-06-04 00:18:00.724629909 +0900 KST m=+199.117396830 [127.0.0.1:40001 127.0.0.1:40002] 127.0.0.1:40001}
INFO[0019] Hot Key : 1_2_3_0
INFO[0019] Block Information - &{{{0 0} 0 0 0 0} 2020-06-04 00:18:00.759413993 +0900 KST m=+199.152180913 [127.0.0.1:40001 127.0.0.1:40002] 127.0.0.1:40001}
INFO[0032] Remove Node - 127.0.0.1:40001
INFO[0032] Sweep Block Phase
INFO[0040] Remove Node - 127.0.0.1:40002
INFO[0040] Sweep Block Phase
```

```
INFO[0013]  - TASK: Heartbeat
INFO[0013]  - TASK: Heartbeat
INFO[0014]  - TASK: Heartbeat
INFO[0014]  - TASK: Heartbeat
INFO[0014] Create Block RPC Call
INFO[0014] Format: /home/wessup/testd1/Block_0.blk
INFO[0014] Create Block RPC Call 00 01 00 02 00 03
INFO[0014] Format: /home/wessup/testd1/Block_1.blk
INFO[0014] Format: /home/wessup/testd1/Block_2.blk
INFO[0014] Format: /home/wessup/testd1/Block_3.blk
INFO[0015]  - TASK: Heartbeat
INFO[0015]  - TASK: Heartbeat
INFO[0016]  - TASK: Heartbeat
```

```
INFO[0029]  - TASK: Heartbeat
INFO[0029]  - TASK: Heartbeat
INFO[0030]  - TASK: Heartbeat
INFO[0030] Garbage Collection For 0 blocks
INFO[0030]  - TASK: GarbageCollection
INFO[0030]  - TASK: Heartbeat
INFO[0031]  - TASK: Heartbeat
```

3rd Iteration

KUORA

No.	Name	Description
1.1	Read File	Client에서 Key를 통해 Read를 요청하였을 때 해당 데이터를 읽어오는가 여부를 확인
1.2	Write File	Client에서 새로운 파일을 만들고 Write 요청을 할 때 해당 데이터가 DataServer에 올바르게 저장되는가 여부를 확인
1.3	Delete File	Client에서 Key를 통해 Delete 요청을 처리 후 해당 데이터가 Garbage Collection 되는지 확인
2.1	Heartbeat	Master가 지속적으로 DataServer로부터 요청을 받아 DataServer가 정상임을 확인할 수 있는지 여부를 확인
2.2	Expire/Migration	Master의 지시에 따라 DataServer가 Expire에 의한 Delete 혹은 Migration을 수행할 수 있는지 여부를 확인
2.3	Garbage Collection	Configure된 주기에 따라 DataServer 내의 실제 Chunk 데이터가 삭제되는지 여부를 확인
2.4	Replication	Client의 Write 후 DataServer에 Chunk 데이터가 복제가 되었는지 확인
2.5	Re-Replication	DataServer 하나를 중지시킨 후 해당 DataServer가 가진 데이터가 다른 서버에 복제되었는지 확인
Q1	Scalability	Master가 새로운 DataNode를 시스템에 포함시킬 수 있다.
Q2	Multiple Client Test	Client를 다수 실행하여 쓰기 및 읽기를 실행하였을 때 데이터의 동일성 및 Key의 유일성을 확인

구현 완료

미구현

일부 구현 / 진행중

Web View

KUORA

KEY LIST

List Key

1: 4_0
2: 0_0
3: 1_0
4: 2_0
5: 3_0

NODE BLOCKS

Get Status

127.0.0.1:40001 - | Block-4 |
127.0.0.1:40002 - | Block-0 | Block-1 | Block-2 | Block-3 |
127.0.0.1:40003 - | Block-0 | Block-1 | Block-2 | Block-3 | Block-4 |

10 0 Read Delete

abvdfesdfsdef

Write

참고문헌

KUORA