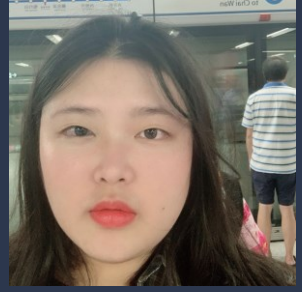
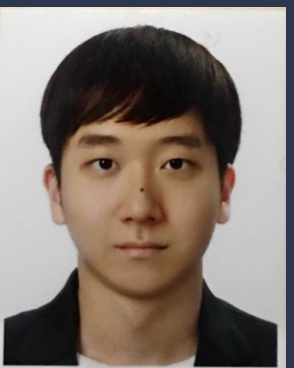


디자인(상위/상세)

KUORA

9 팀 (서지원, 홍나리) I 졸업프로젝트 1 I 유준범 교수님

2020.04.25



Requirements

KUORA

• Functional Requirements

1. Client Side

1.1. Read File
- Client가 Master에게 Key를 통하여 File에 해당하는 Chunk의 위치를 요청한다.
- Client는 해당 정보를 통하여 DataServer로부터 파일을 읽는다.

1.2. Write File

1.2.1. Create
- Client가 Master에 파일의 Create를 요청한다.
- Master는 논리적인 File에 해당하는 Key를 만들고 Chunk의 생성을 DataServer에 요청한다.
- Master은 Key와 Chunk의 정보를 저장하고 생성된 Key를 Client에게 전달한다.

1.2.2. Write Chunk
- Client가 Key를 통해 Master에 Write를 요청한다.
- Master는 Chunk의 위치와 Primary와 Secondary 서버의 주소를 전달한다. (기본 복제본은 3개이므로 3개의 서버의 위치를 전달한다.)
- Client는 해당 정보를 통하여 DataServer의 Chunk에 정보를 write한다.
- 2.4 (Replication) 작업이 수행된다.

1.3. Delete File
- Client가 File Key를 통하여 Master에게 요청한다.
- Master는 해당 Key의 정보를 제거하고 해당 Chunk를 Garbage 리스트에 넣고 DataServer에게 전달한다.

2. System Side (Master + DataServer)

2.1. Heartbeat
- DataServer는 Master에게 자신의 서버 주소와 정상 동작 여부를 전송한다. (주기: 400 ~ 800ms)
- Master는 Heartbeat가 이루어지는 서버의 정보를 저장한다.

2.2. Expire/Migration
- 처음 저장되는 모든 File은 Hot Storage에 저장된다.
- Hot Storage의 Hot File의 Expire Time이 부여되며 이 시간이 지날 경우 Cold Storage로의 Migration이 시행된다.

2.3. Garbage Collection
- DataServer는 주기적으로 Master에서 전달된 Garbage 리스트의 Chunk에 대해서 가비지 컬렉션을 수행한다. (주기: 1 day – Configurable)

2.4. Replication
- 처음 파일이 생성되고 데이터가 쓰여질 때, Client는 Master가 지정한 Primary에 데이터를 쓴다.
- Primary로 선정된 DataServer는 그 외의 Secondary 서버들에게 데이터를 전달한다.

2.5. Re-Replication
- Master는 HeartBeat(2.1)을 통해 DataServer의 상태를 점검할 수 있다.
- DataServer가 장기간 Heartbeat를 보내지 못했을 경우 (약 10번의 Heartbeat) 해당 DataServer가 장애가 난 것으로 간주한다.
- 장애가 난 DataServer(Fault)가 가진 Chunk들의 데이터를 가진 다른 DataServer(D1)가 Chunk를 지니지 않은 다른 DataServer(D2)에게 데이터를 전달하여 복제본의 수(3)를 유지한다.
- 작업이 끝나면 Master는 DataServer(Fault)의 정보를 제외한다.

• Quality Requirements

Q1. Scalability

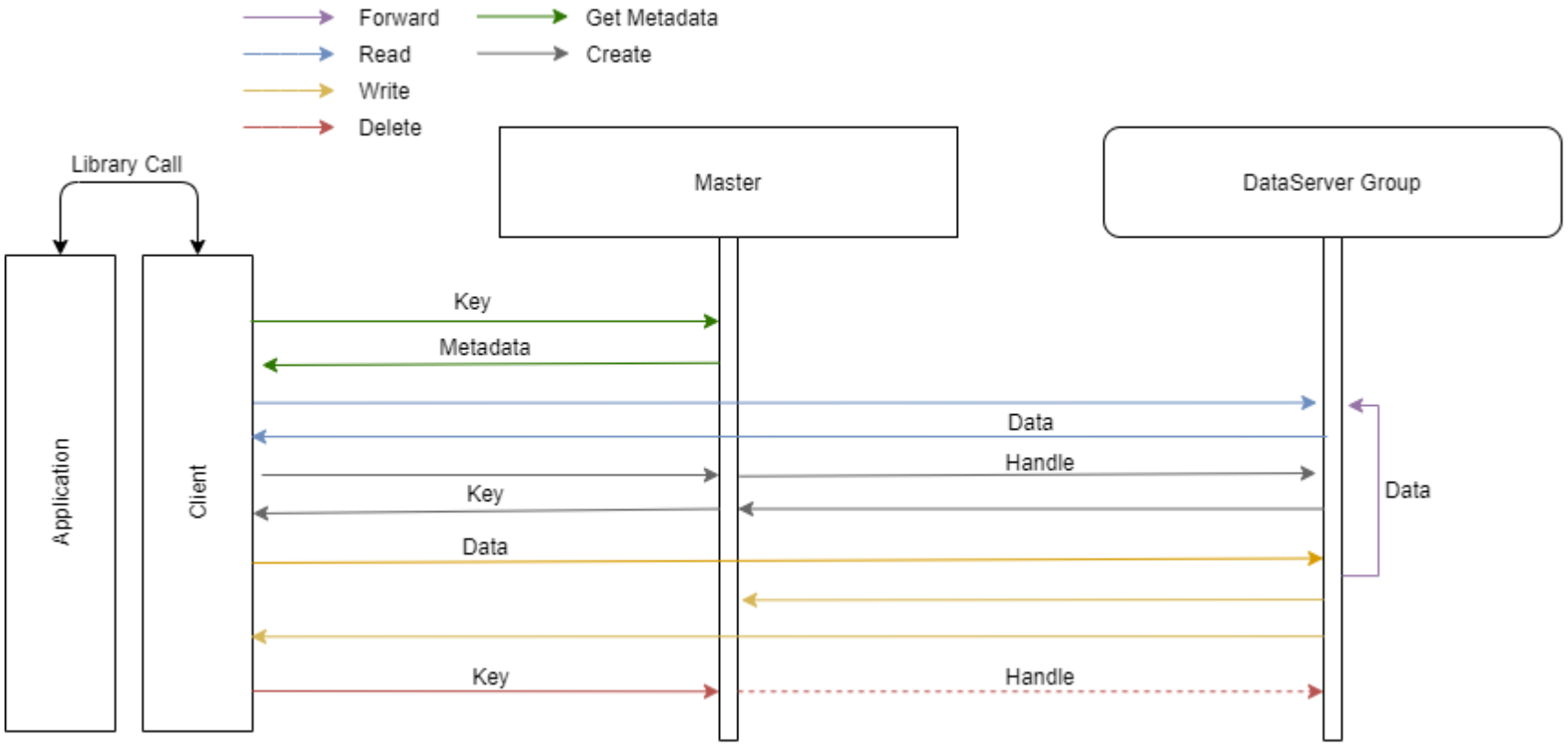
- Master는 새로운 DataServer가 실행될 경우 이를 시스템에 포함시킬 수 있다.

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.

Design

System Sequence Diagram



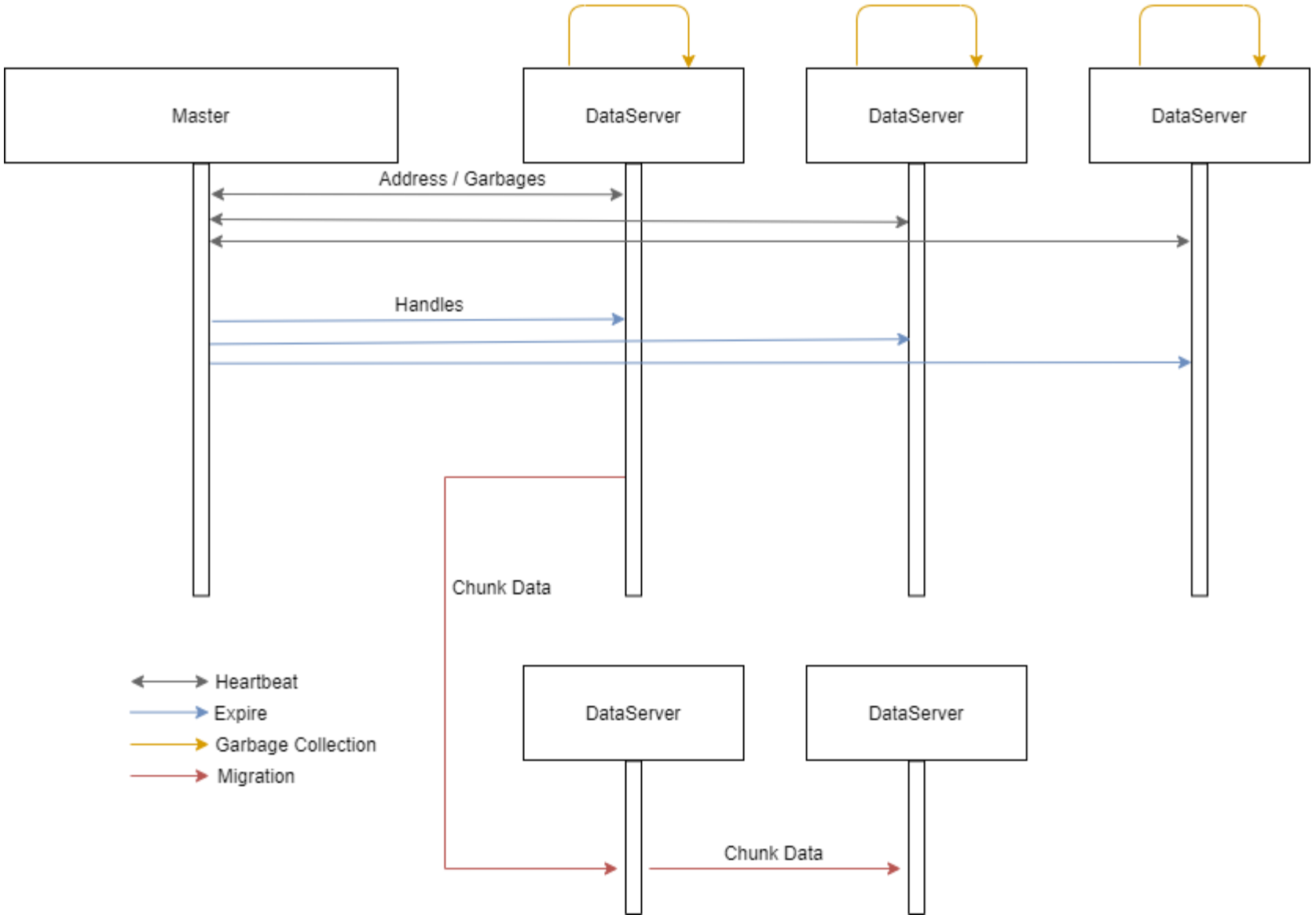
Read
1. Exchange Metadata
2. Read Data

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

Delete
1. Delete Request

Design

System Sequence Diagram



Heartbeat

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

Expire/Migration

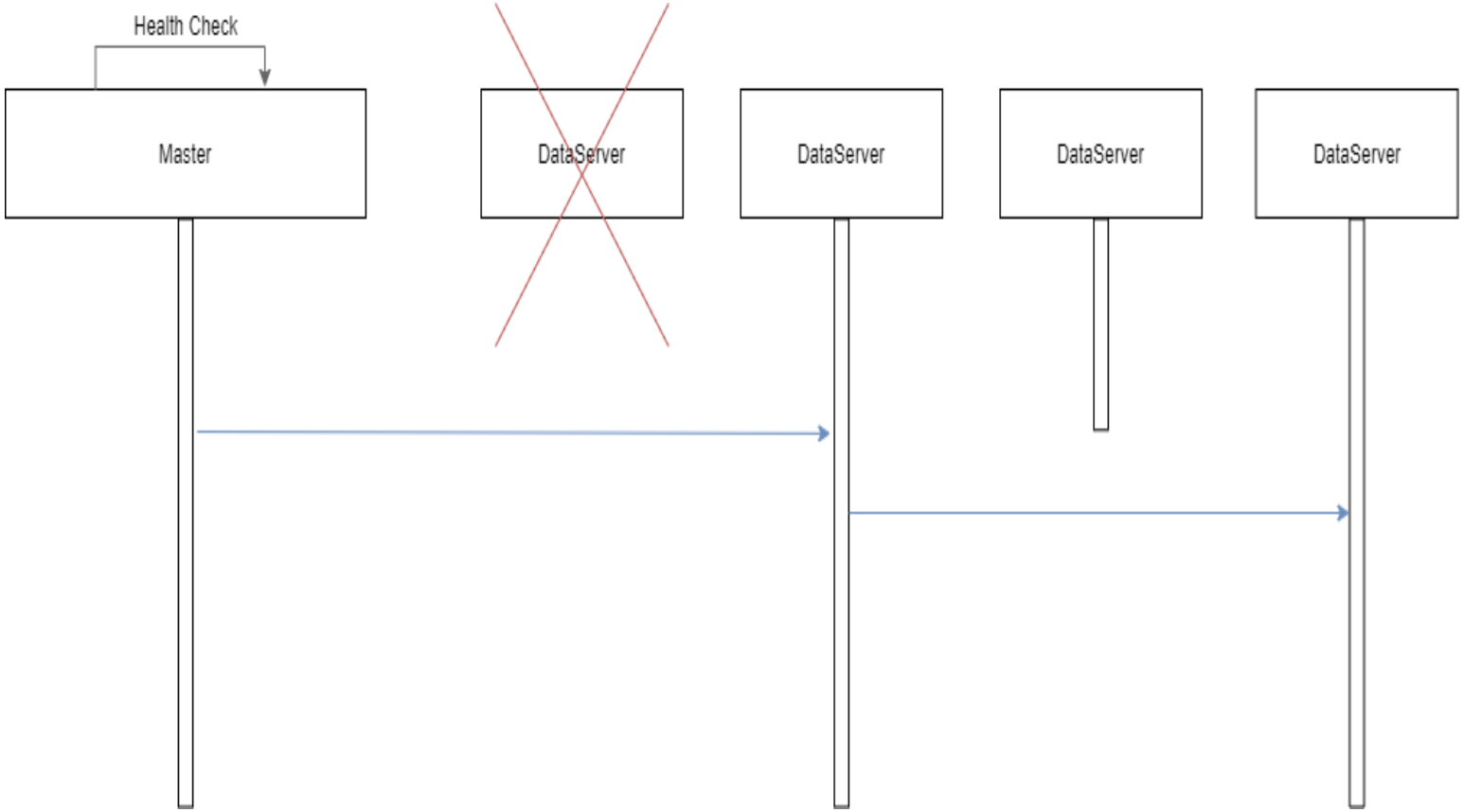
1. Master는 주기적으로 Chunk의 Expire 여부 체크
2. Expired Chunks를 Migration하도록 DataServer에 지시
3. DataServer가 Migration 수행

Garbage Collection

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

Design

System Sequence Diagram



IF.

DataServer 중 하나가 비정상적으로 Shutdown 되었을 시

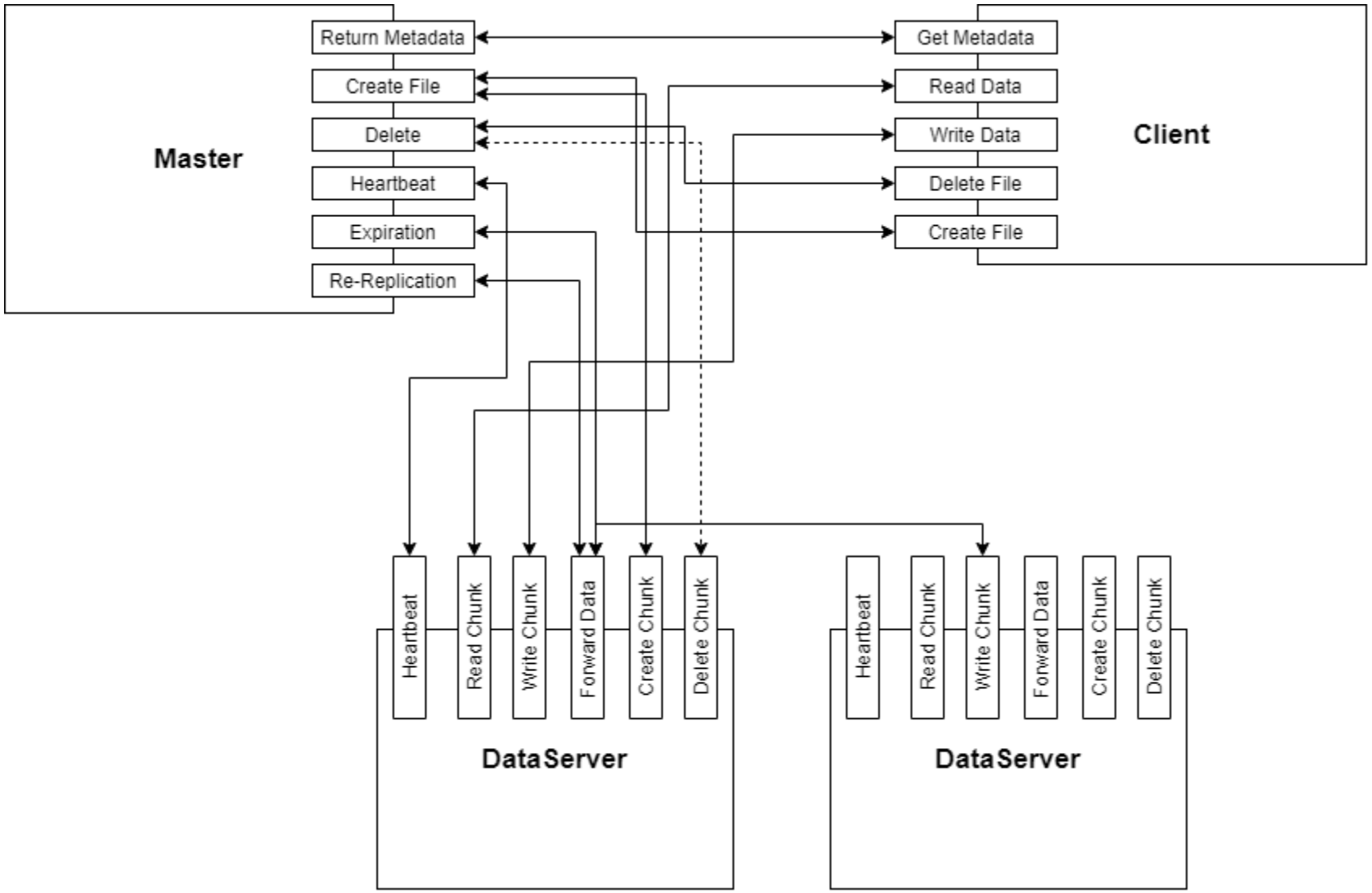
1. Heartbeat의 전송이 끊기며 Master가 Health Check를 통해 장애(Fault) 파악

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

3. DataServer가 이를 수행

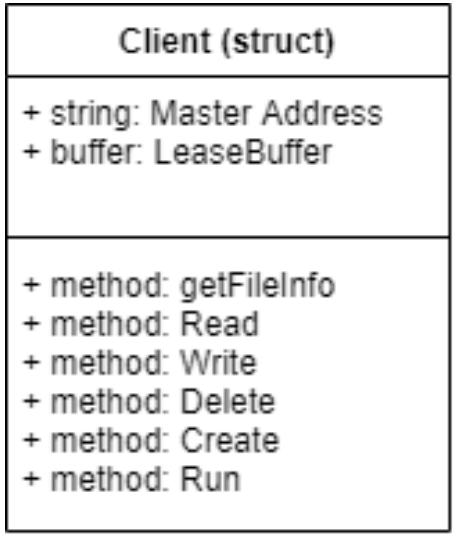
Design

High-Level Interfaces



Design

Low-Level Design



Client

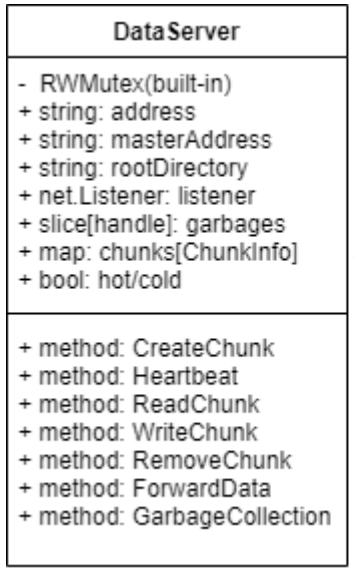
1.1 getFileInfo
Key를 통해 File의 Metadata 얻을 수 있음.

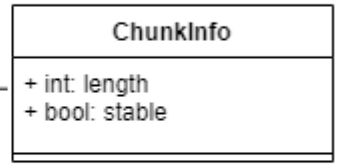
1.2 Read
Chunk를 Read

1.3 Write
Chunk에 data를 Write

1.4 Delete
Key를 통해 File을 지우도록 요청.

1.5 Create
Master에 File을 생성하도록 요청.





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.

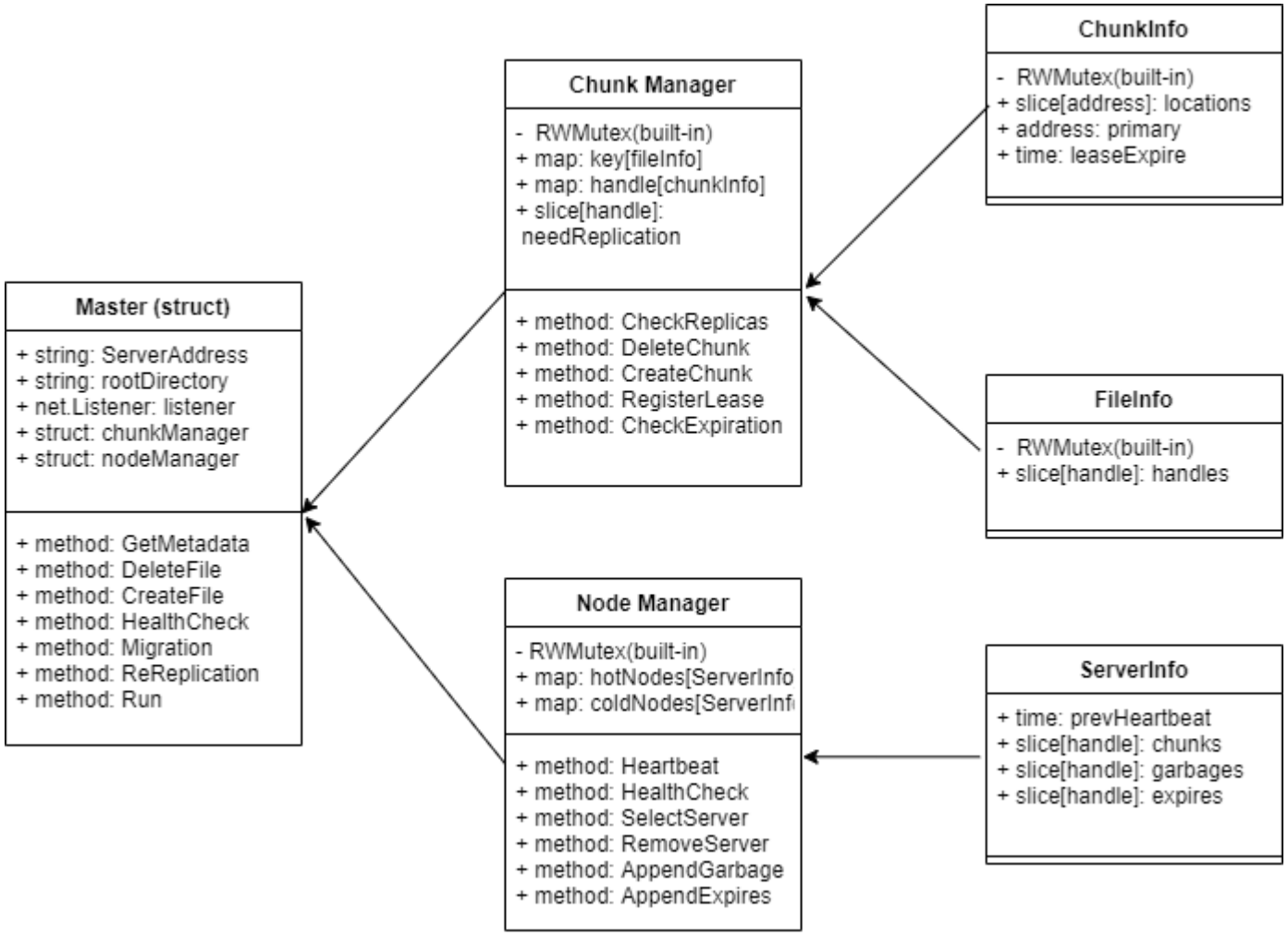
2.5 RemoveChunk
Chunk를 삭제

2.6 ForwardData
다른 DataServer로 Chunk data 전송.

2.7 Garbage Collection
주기적으로 실행

Design

Low-Level Design



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

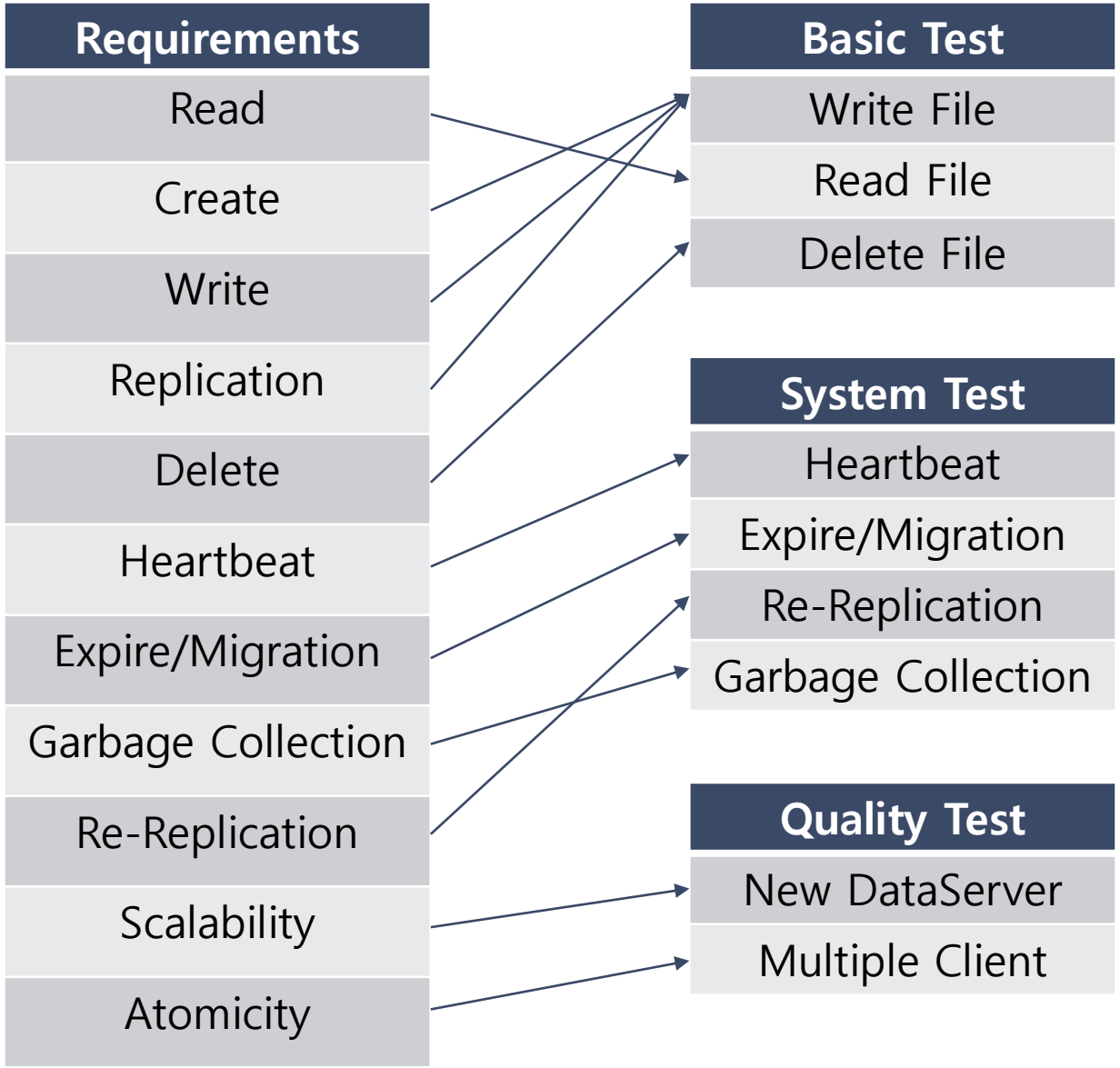
3.4 HealthCheck
Heartbeat를 통하여 각 서버의 상태 체크

3.5 Migration
Expired Chunks를 파악하여 Migration을 지시

3.6 ReReplication
DataServer에 ReReplication을 지시

Design

Traceability Matrix



1. Basic Test
   - Write File

   Test Code를 통하여 Client 요청으로 File(Dummy Data) 쓰기 테스트
   - Read File

   Client 요청으로 File을 읽어 Dummy Data와 일치하는지 확인
   - Delete File

   Client 요청으로 File 지우기 테스트

2. System Test
   - Heartbeat

   Master와 DataServer 간의 데이터 교환
   - Expire/Migartion

   Master가 Expire된 Chunk에 대해서 DataServer에 Migration을 지시
   - Re-Replication

   DataServer를 하나 Shutdown한 후 Chunk의 데이터가 복사되는지 확인
   - Garbage Collection

   config된 tick마다 DataServer내의 Garbage가 지워지는지 확인

3. Quality Test
   - New DataServer

   새로운 DataServer 추가하여 정상 작동하는지 테스트
   - Multiple Client

   다수의 thread를 통해 요청하여 정상 작동하는지 테스트

Design

Traceability Matrix

Basic Test	High-Level	1.1	1.2	1.3	1.4	1.5	2.1	2.2	2.3	2.4	2.5	2.6	2.7	3.1	3.2	3.3	3.4	3.5	3.6
Write File	Exchange Metadata	O												O					
	Create					O	O									O			
	Write			O						O									
	Replication																		
Read File	Exchange Metadata									O		O		O					
	Read	O																	
Delete File	Delete		O						O						O				
					O						O		O						

System Test	High-Level																		
Heartbeat	Heartbeat							O									O		
Expire/Migration	Scheduled Expire																		
	Migration Instruction						O	O		O		O						O	
Re-Replication	Health Check							O									O		
	Re Replication						O			O		O							O
Garbage Collection	Scheduled GC										O		O				O		

Test Case

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