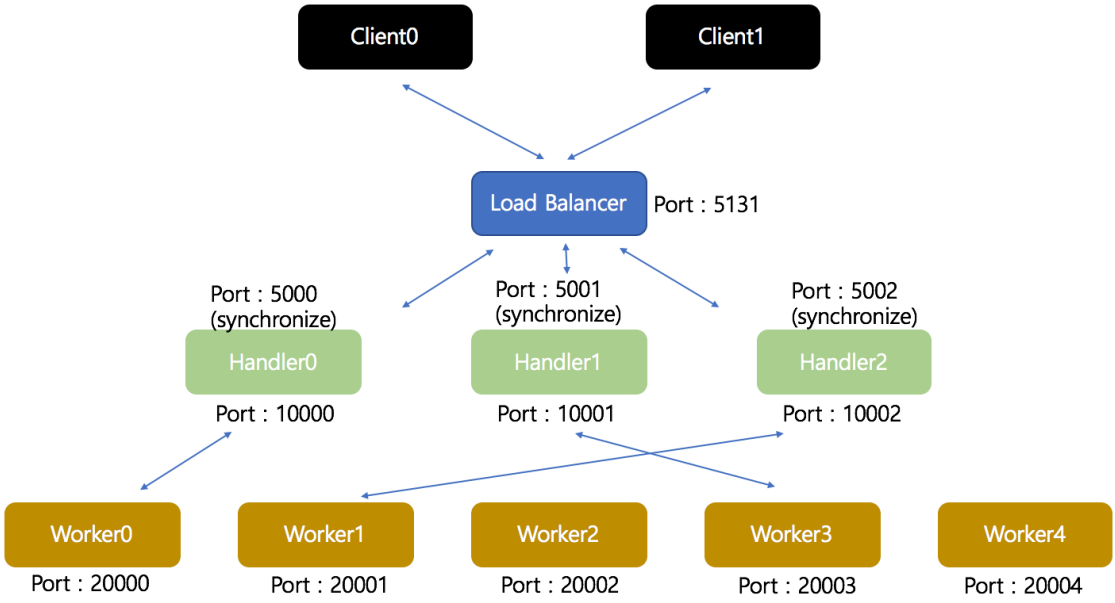
Programming Assignment 1

20173255 방지환

1. 시스템 구조

이 시스템 구조는 각 모듈이 어떻게 연결되어 있는 지, 그리고 연결되어 있는 port 번호를 보여주고 있다. 이번 숙제에서 모든 모듈의 IP 주소는 localhost 인 127.0.0.1 로 하였다. (단, Load Balancer 는 client 에서 처음에 지정할 수 있게 하였다.)



2. 프로토콜

이 시스템은 "MessageProtocol"와 "MessageHD2WK"라고 불리는 두 개의 프로토콜이 존재한다. MessageProtocol 은 client 와 load balancer 간, load balancer 와 handler 간의 통신에 쓰이는 프로토콜이다. 그리고 MessageHD2WK 는 handler 와 worker 간의 통신에 쓰이는 프로토콜이다. 아래는 프로토콜의 구조 및 프로토콜이 사용되는 함수에 대해 설명한다.

2.1. MessageProtocol

MessageProtocol structure{

int clientID

int sequence

short command

short code

short keyLength

short valueLength

String key
String value
}

MessageProtocol 은 send, receive, messageUpdate, ackProcess, print 총 5 가지의 함수가 존재한다. 함수에 대한 설명은 "MessageProtocol.java"에 이미 코멘트로 명시하였으므로 본 보고서에서는 생략하도록 한다.

2.2. MessageHD2WK

MessageHD2WK structure{
int command
int hashValue
String key
String value
}

MessageHD2WK 는 messageUpdate, send, receive, print 총 4 가지의 함수가 존재한다. 함수에 대한 설명은 "MessageHD2WK.java"에 이미 코멘트로 명시하였으므로 본 보고서에서는 생략하도록 한다.

3. 모듈

이번 시스템은 client, load balancer, handler, worker 총 4 가지의 모듈이 존재한다. 이번 장에서는 각각의 모듈에서 하는 일 및 시스템의 동작 원리에 대해 설명하고자 한다.

3.1. Client

Client 는 아래와 같은 순서로 정보를 받고, data 저장, key 를 통한 value 가져오기, key 를 통해 저장된 key/value 값 지우기 총 3 가지의 명령 중 하나를 load balance 에게 전달한다.

1. connect [LB IP] : load balance 의 IP 주소를 지정한다.
2. set [client number] : client ID 를 지정한다.
3. put/get/del : 계속해서 load balance 에 instruction 을 보내고, 그에 대한 ack 을 받아 잘 처리되었는 지의 여부를 확인한다.

또한, 한 instruction 을 수행한 이후에는 sequence 의 값을 1 씩 증가시킨다.

3.2. Load Balancer

Load balancer 는 client 로부터 오는 정보를 handlers 에게 공평하게 전해주는 역할을 한다 (Round Robin : handler0 -> handler1 -> handler2 -> handler0 -> ...). 또한, handler 로부터 오는 ack 을 받아 client 에게 ack 을 전송하는 일도 한다.

CLI 에서 "list"와 같은 명령어를 처리하기 위해서 이는 다른 LBCLI 라는 thread 를 이용하여 independent 하게 진행되도록 하였다. 또한, LBCLI 와 main 과 서로 handler 에게 주

는 request 수를 공유하기 위해서 SharedArea 라는 class 를 만들었다.

3.3. Handler

Handler 는 load balancer 에게 온 정보를 가지고 instruction 을 해석한다. 그리고 각각의 instruction(put, get, del)에 맞게 행동한다. Put 인 경우에는 동기화된 리스트에 이미 저장이 되어있는 지를 확인한 후, 없으면 MessageHD2WK (key 의 hash 값, key, value)로 worker 에 보낸다. Worker 로부터 ack 을 받은 후에 load balance 에게 ack 을 보낸다. Get 의 경우에도 동기화된 리스트를 보고, 특정 worker 에 key 가 저장되어 있다면, worker 에게 value 값을 요청하고, ack 과 함께 value 값을 받는다. 그 이후, value 값을 가지고 load balance 에게 보내준다. 이 후에 역시 put 과 마찬가지로 ack 을 보낸다. Del 의 경우에는 동기화된 리스트를 보고, 존재하면 리스트에서 제외한다. 또한, worker 로 MessageHD2WK 를 보내어 삭제되어야 할 key 가 저장되어 있는 worker 에서 data 를 지우도록 한다. 이 때 worker 로부터 ack 을 받고, 그 이후에 ack 을 load balance 에게 보낸다. 특정 worker 를 정하는 방법은 key 의 hashvalue 를 5 로 나눈 나머지를 worker id 로 한다.

hashValue 의 경우에는 joaat_hash 라는 함수를 사용하였고, 이는 https://en.wikibooks.org/wiki/Algorithm_Implementation/Hashing#Jenkins_one-at-a-time_hash 사이트를 참조하였다.

동기화와 같은 경우에는 put, del 을 한 경우에만 해당한다. Put, del 을 하게 되면 동기화리스트의 element 들이 변하기 때문에 다른 handler (다른 process)에게 리스트 정보를 업데이트할 수 있도록 해줘야 한다. 이를 해결하기 위해 Synchronization 이라는 server thread 를 만들었고, put/del 을 처리하는 handler 가 다른 handler 에게 socket 을 통해 HDelement (command(put/del), hashValue, worker ID)를 전해줌으로써 리스트들을 서로 동기화시킨다. (port 번호는 위의 그림에 명시되어 있다.)

HDCLI 도 3.2 와 마찬가지로 thread 를 만들어 main 과 독립적으로 진행된다. 이는 CLI 에서 "list", "show"를 통해 현재 어느 worker 가 어떠한 hash value 를 가지고 있는지를 볼 수 있다. 만약 동기화가 제대로 이루어졌다면, 3 개의 handler 모두 같은 결과를 보여줄 것이다.

3.4. Worker

Worker 는 handler 로부터 오는 instruction 을 처리한다. Put 의 경우에는 worker 의 wklist 에 (hashvalue, key, value) 형태로 저장한다. 이 후에는 handler 에게 ack 을 보내준다. Get 의 경우에는 hashvalue 를 통해 key 와 value 를 찾은 후, value 값과 함께 handler 에게 ack 을 보내준다. Del 의 경우에는 wklist 에서 해당하는 hashvalue 의 list element 를 지운다. 그 이후에는 handler 에게 다시 ack 을 보낸다.

3.2, 3.3 과 같이 WKCLI 라는 독립된 thread 를 만들어 CLI 에 "list", "show" 명령어를 사용할 수 있도록 하였다. 이 명령어를 통해서 현재 worker 에 저장되어 있는 값들이 무엇이 있는지를 보여준다.

4. LOG file

Log 파일은 "[module name].log"형태로 만들었으며, 모든 받은 정보들을 받은 시간과 함께 print 하도록 하였다. (단, 프로세스가 실행되기 전에 이름이 같은 log file 이 있다면, 이를 덮어쓰도록 처리하였다.)

Client : Loadbalance 로부터 오는 MessageProtocol ack

LoadBalance : Client 로부터 오는 MessageProtocol instruction, Handler 로부터 오는 MessageProtocol ack

Handler : LoadBalance 로부터 오는 MessageProtocol instruction, worker 로부터 오는 MessageHD2WK ack

Worker : Handler 로부터 오는 MessageHD2WK

5. Conclusion

이 숙제를 통하여서 데이터센터에서 어떻게 데이터들을 가져오고(get), 저장하고(put), 없애는지(del)를 알게되었다. 간단한 architecture 이지만, handler 에서 데이터 공유의 중요성 및 key/value scheme 에 대해 완벽히 숙지할 수 있는 좋은 경험이었다.

6. Reference

1. hash value function : https://en.wikibooks.org/wiki/Algorithm_Implementation/Hashing#Jenkins_one-at-a-time_hash
2. instructions.pdf
3. PA1(updated).pdf