

## Lab5. Network lab

컴퓨터공학부 2022-19510 오정윤

### 1. TCP/IP network environment

우선 서버에서는 `start_server()`에서 listen socket을 만들고 client의 request를 기다린다. 이를 위해 `getsocklist()`에 `AF_INET`, `SOCK_STREAM` 변수를 인자로 넘겨주어 IPv4, TCP 환경에서의 listen용 socket을 찾고 while문을 통해 `bind()`와 `listen()`이 성공하는 listen socket을 만든다. 또한 while문을 통해 계속 listen socket으로 오는 client request를 받아 최대 고객 수를 넘지 않는지 "`if (server_ctx.total_queueing >= CUSTOMER_MAX)`"를 통해 검사하고, 넘는다면 while문을 continue한다. 넘지 않는다면 `total_queueing`을 증가시키고 `serve_client` thread를 만든다. 이때 인자로 넘겨주는 `thread_client_fd`는 `malloc()`을 통해 고정한다.

동시에 클라이언트에서는 `main()`에서 미리 n개의 client task thread를 만든다. 이 `thread_task`는 서버와 마찬가지로 `getsocklist`를 통해 리스트를 가져오고 `connect()`가 성공하는 socket을 찾는다.

`serve_client` thread와 `thread_task` thread는 `put_line()`, `get_line()`을 통해 메시지를 전달한다. 특히 `serve_client`에서는 `thread_task`에서 오는 request를 `get_line()`하여 여러 개의 order로 나눈다. 이렇게 나눈 order를 `issue_orders()`를 통해 enqueue한 후 `main()`에 의해 작동되고 있는 `kitchen_task` thread에 의해 dequeue되도록 한다.

### 2. Atomicity between threads

같은 `ordre_list`에 속한 order들이 `remain_count`, `cond`, `cond_mutex`를 공유하고 있기 때문에 `pthread_mutex_lock()`을 통해 일관성을 보장해야 했다. `skeleton`에서는 `issue_orders()`를 실행할 때 노드를 enqueue하면서 서버 차원의 뮤텁스를 사용하여 링크 도중의 데이터 일관성을 보장하고 있다. 또한 `order_list`의 `remain_count`가 전부 동일하기 때문에 이를 보호하는 것이 중요했다. 이를 위해 `serve_client` thread에서는 첫번째 order의 `remain_count`가 양수일 때까지, 즉 order가 남아있을 때까지 `pthread_cond_wait()`을 통해 `serve_client` thread 작동을 기다렸다. 또한 `kitchen_task` thread에서는 `make_burger()`를 실행하고 `remain_count`를 감소시키는 과정 전후로 뮤텁스를 추가하여 `remain_count`의 값을 보호했다. 이뿐만 아니라 주어진 `skeleton`에서는 `get_order()`와 `order_left()`, 즉 dequeue할 때와 남은 order 수를 살필 때에도 server context의 뮤텁스를 사용하여 데이터를 보호하고 있다. 또한 통계를 위한 `total_burgers[]`를 증가시킬 때, customer ID를 정하기 위해 `total_customer`를 증가시킬 때, 주문 완료 후 `total_queueing`을 감소시킬 때 데이터를 보호하고 있다.

### 3. optimization

앞서 언급했던 `kitchen_task`에서의 mutex 활용에서, global variable인 `kitchen_mutex`를 사용하면 `kitchen` thread 간의 지연이 발생하여 최적화를 위해서는 이를 사용하지 않아야 했다. 따라서

make\_burger()과 remain\_count 감소하는 과정을 order\_list들이 공유하는 order->cond\_mutex를 사용하여 lock, unlock을 했다. remain\_count를 보호하는 본래 목적이 같은 order\_list 내에서 remain\_count를 공유하기 때문에 이를 이용하는 것이 더 효율적이다. 실제로 kitchen\_mutex를 사용하였을 때는 reference와 비슷한 성능을 보였는데, cond\_mutex를 사용한 결과 시간을 절약할 수 있었다.

```
sysprog@sysprog-lab:~/sysprog_lab5$ time ./client 5
[Thread 140386549778176] From server: Welcome to McDonald's, customer #10
[Thread 140386549778176] Ordering 3 burgers
[Thread 140386549778176] To server: Can I have bulgogi chicken cheese burger(s)?
[Thread 140386558170880] From server: Welcome to McDonald's, customer #11
[Thread 140386558170880] Ordering 3 burgers
[Thread 140386558170880] To server: Can I have bulgogi cheese bulgogi burger(s)?
[Thread 140386566563584] From server: Welcome to McDonald's, customer #12
[Thread 140386566563584] Ordering 3 burgers
[Thread 140386566563584] To server: Can I have chicken bigmac cheese burger(s)?
[Thread 140386574956288] From server: Welcome to McDonald's, customer #13
[Thread 140386574956288] Ordering 3 burgers
[Thread 140386574956288] To server: Can I have cheese chicken bulgogi burger(s)?
[Thread 140386583348992] From server: Welcome to McDonald's, customer #14
[Thread 140386583348992] Ordering 3 burgers
[Thread 140386583348992] To server: Can I have chicken bulgogi bulgogi burger(s)?
[Thread 140386549778176] From server: Your order(bulgogi chicken cheese) is ready! Go
odbye!
[Thread 140386566563584] From server: Your order(chicken bigmac cheese) is ready! Goo
dbye!
[Thread 140386558170880] From server: Your order(bulgogi cheese bulgogi) is ready! Go
odbye!
[Thread 140386574956288] From server: Your order(cheese chicken bulgogi) is ready! Go
odbye!
[Thread 140386583348992] From server: Your order(chicken bulgogi bulgogi) is ready! G
oodbye!

real    0m4.647s
user    0m0.000s
sys     0m0.004s
```

그림 1. Client 5일 때 최적화 코드의 실행 시간

```
sysprog@sysprog-lab:~/sysprog_lab5/reference$ time ./client 5
[Thread 140356225074944] From server: Welcome to McDonald's, customer #10
[Thread 140356225074944] Ordering 3 burgers
[Thread 140356225074944] To server: Can I have cheese bulgogi chicken burger(s)?
[Thread 140356216682240] From server: Welcome to McDonald's, customer #11
[Thread 140356216682240] Ordering 3 burgers
[Thread 140356216682240] To server: Can I have bigmac bigmac chicken burger(s)?
[Thread 140356233467648] From server: Welcome to McDonald's, customer #12
[Thread 140356233467648] Ordering 3 burgers
[Thread 140356233467648] To server: Can I have bulgogi bulgogi chicken burger(s)?
[Thread 140356241860352] From server: Welcome to McDonald's, customer #13
[Thread 140356241860352] Ordering 3 burgers
[Thread 140356241860352] To server: Can I have cheese cheese bulgogi burger(s)?
[Thread 140356250253056] From server: Welcome to McDonald's, customer #14
[Thread 140356250253056] Ordering 3 burgers
[Thread 140356250253056] To server: Can I have chicken bulgogi cheese burger(s)?
[Thread 140356225074944] From server: Your order(cheese bulgogi chicken) is ready! Go
odbye!
[Thread 140356216682240] From server: Your order(bigmac bigmac chicken) is ready! Goo
dbye!
[Thread 140356233467648] From server: Your order(bulgogi bulgogi chicken) is ready! G
oodbye!
[Thread 140356241860352] From server: Your order(cheese cheese bulgogi) is ready! Goo
dbye!
[Thread 140356250253056] From server: Your order(chicken bulgogi cheese) is ready! Go
odbye!

real    0m15.024s
user    0m0.004s
sys     0m0.000s
```

그림 2. Client 5일 때 reference 코드의 실행 시간

```

dbye!
[Thread 140158803891968] From server: Your order(chicken bigmac chicken) is ready! Go
odbye!
[Thread 140158812284672] From server: Your order(bigmac bigmac bulgogi) is ready! Goo
dbye!
[Thread 140158820677376] From server: Your order(bigmac bulgogi cheese) is ready! Goo
dbye!
[Thread 140158829070080] From server: Your order(chicken chicken chicken) is ready! G
oodbye!
[Thread 140158837462784] From server: Your order(bulgogi bulgogi bulgogi) is ready! G
oodbye!

real    0m4.500s
user    0m0.000s
sys     0m0.004s

```

그림 3. Client 10일 때 최적화 코드의 실행 시간

```

odbye!
[Thread 140305035699968] From server: Your order(bigmac chicken cheese) is ready! Goo
dbye!
[Thread 140305044092672] From server: Your order(bigmac cheese bigmac) is ready! Good
bye!
[Thread 140305052485376] From server: Your order(bulgogi chicken cheese) is ready! Go
odbye!
[Thread 140305027307264] From server: Your order(bigmac bigmac bigmac) is ready! Good
bye!
[Thread 140305060878080] From server: Your order(chicken bigmac chicken) is ready! Go
odbye!
[Thread 140305069270784] From server: Your order(bigmac bigmac cheese) is ready! Good
bye!
[Thread 140305077663488] From server: Your order(bigmac chicken chicken) is ready! Go
odbye!
[Thread 140305086056192] From server: Your order(bulgogi bigmac cheese) is ready! Goo
dbye!
[Thread 140305094448896] From server: Your order(bulgogi bigmac cheese) is ready! Goo
dbye!

real    0m30.302s
user    0m0.005s
sys     0m0.000s

```

그림 4. Client 10일 때 reference 코드의 실행 시간