1.time.x를 이용하여 rpc함수들을 정의하고 rpcgen을 이용하여 클라이언트 스텁과 서버 스텁을 만들었다. time\_client.c는 rpc클라이언트 프로그램이며 time\_server.c와 time\_svc.c는 서버 프로그램이다. 클라이언트는 get\_time을 호출하고, 실행 시에 입력 받은 숫자만큼 쓰레드를 생성하여 각각의 쓰레드가 임의의 숫자만큼 delay\_time을 호출한다. server는 처음 실행될 때 gettimeofday를 이용하여 초기 시간을 저장하고, 클라이언트가 get\_time을 호출할 때마다 경과 시간을 계산하여 반환한다. 클라이언트가 delay\_time을 호출하면 함수인자로 넘어온 숫자만큼 초기 시간을 증가시켜 서버 시간을 delay시킨다. delay 중간에 쓰레드들 간의 간섭이 없도록 클라이언트와 서버 모두 mutex를 이용하여 동기화하였다.

2.my\_rwlock.c에는 pthread\_mutex와 condition variable을 이용한 read-write lock을 구현하였다. pth\_ll\_rwl은 기존의 pthread rwlock을 사용하는 프로그램이고, pth\_ll\_own은 my\_rwlock.c의 rwlock을 사용하는 프로그램이다. 이 두 프로그램을 비교한 결과는 다음과 같다.

아래의 실험은 쓰레드의 개수는 100개, 리스트에 삽입한 key의 개수는 10000개, 실행한 명령의 수는 100000개로 고정하고 각각 명령의 비율을 조정하며 실험한 결과이다. 각각 member : insert : delete의 비율이 0.5 : 0.25 : 0.25, 0.7 : 0.15 : 0.15, 0.9 : 0.05 : 0.05이다.

pth\_ll\_rwl과 pth\_ll\_own 모두 read 명령인 member의 비율이 높아짐에 따라서 성능이 올라가는 것을 통해 read-write lock의 효과를 볼 수 있다.

3.주어진 micor\_httpd를 이용하여 멀티쓰레드 http서버를 구현하였으며 이를 실험하기 위한 client도 멀티쓰레드를 이용하여 구현하였다. client는 실행시에 입력한 개수만큼 쓰레드를 생성하여 서버로 요청을 보낸다. 멀티 쓰레드 서버의 성능을 비교하기 위하여 서버의 쓰레드 개수에 따른 실행 시간을 비교하였다.

서버의 쓰레드가 증가함에 따라서 결과 시간이 감소해야 할 것 같았지만 오히려 증가하는 것을 볼 수 있다. 이러한 이유로는 서버에서 queue에 요청을 넣는 것은 단일 쓰레드가 하기 때문이며 thread들 간의 동기화와 pthread\_cond의 wait/signal에서 오히려 오버헤드가 발생하는 것으로 추측된다.