Pthread를 이용해서 행렬의 곱셈 병렬로 만들기

2009003039

조현석

POSIX thread를 이용하여 순차적으로 실행되던 행렬의 곱셈을 여러 개의 쓰레드로 나누어 실행 할 수 있게 만들었다. 전처리문으로 최대한 앞에 행렬의 길이를 성능향상을 측정하기에 충분히 큰 값으로 #define VECLEN 100000000을 #include 다음에 써주었다. 몇개의 쓰레드를 생성 할 지 #define THREADLEN 2 에서 정의 해 주었다.

DOTDATA.sum에 동시에 접근하는 것을 방지하기 위해 mutex1을 선언해 주었고 부분합을 구한 다음 락을 걸고 전체 합계에 더한다.

쓰레드 함수인 dotprod의 매개변수로 시작 인덱스를 받는다. 끝 인덱스는 자동으로 계산한다.

mysum은 double형이므로 0.0을 넣어 준다.

Time 프로그램을 이용하여 GCC 최적화 옵션과 쓰레드 개수별 성능을 비교 해 보았다. (단위는 초)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Serial O0 | Serial O3 | 2thread O0 | 2thread O3 | 4thread O0 | 4thread O3 | 8thread O0 | 8thread O3 |
| 3.499 | 2.965 | 3.002 | 2.814 | 2.986 | 2.495 | 3.05 | 2.471 |
| 3.318 | 2.912 | 2.997 | 2.472 | 2.985 | 2.487 | 2.968 | 2.475 |
| 3.213 | 2.498 | 2.985 | 2.505 | 2.972 | 2.493 | 2.982 | 2.467 |
| 3.203 | 3.171 | 3.03 | 2.495 | 2.971 | 2.628 | 2.965 | 2.521 |
| 3.30825 | 2.8865 | 3.0035 | 2.5715 | 2.9785 | 2.52575 | 2.99125 | 2.4835 |

성능 향상이 있긴 하였지만 미미한 정도라고 생각한다. 이는 전체 프로그램에서 병렬화 된 부분이 차지하는 부분의 전체에 중 일부이고 메모리를 할당하고 해제하거나 문자를 출력하는 부분에서 시간이 오래 걸렸기 때문이라고 생각한다. 메모리 대역폭이 CPU의 계산 속도에 못 미치는 것도 성능향상이 적은 이유일 것이다.