확장형 고성능 컴퓨팅 hw3

2022-25448
박기범

1.

a)

Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz

로

Cascade lake 아키텍쳐이다

b)

각각 두 소켓을 합쳐서 소켓 1/32/44 MB이다.

소켓당으론 0.512/16/22MB이다.

그리고 16*2코어이므로

각각

L1:32KB 코어당이고

L2:1MB 코어당

L3:쉐어되는 22MB 캐쉬 (소켓당)

c)

모두 write back cache 이다.

d)

한 캐쉬라인은 64B

L1 : 8way set associative cache 이고 32kB/64Byte=512 512/8way= 64 line

L2: 16way set associative cache 이고 1MB/64B/16way = 1024 line

L3: 11way set associative cache 22*1MB/64B/22way = 65536line

2.
과제 2와 거의 같은 방식입니다. 캐시의 로컬리티를 높이기 위하여 모든 m,n,k의 문제를 쪼개어서 문제를 할당하였고 추가적인 병렬 연산으로 16개의 플롯을 벡터로 연산하기 위하여 AVX512를 사용하였습니다.

#pargam omp와
omp_set_num_threads(num_threads);
를 이용하여 이루어 집니다. 그리고 런타임 중에 필요한 수만큼의 쓰레드가 자동으로 생성된다고 이해하였습니다.

1: 23.9
16: 149.11
32: 220.94
48: 130.7
64: 97.95
80: 165.1
96: 274.1
112: 288.4
128: 298.4
144: 207.4
160: 192.8
176: 184.2
192: 244.0
208: 152.24
224: 213.92
240: 62.6
256: 300.1

최적의 성능은 256 128 등에서 발생 하였습니다. 아마 제코드가 타일링 하는과정에서 비효율적으로 짜인 부분이 존재하여 비 일관적인 모습을 보이는 부분이 있어 보입니다. 32 쓰레드 이상으론 코어를 실질적으로 대부분 활용한다고 볼 수 있

어 보입니다. 그러나 64등에서 갑자기 떨어지는 이유에 대해선 1024 1024 1024로 제가 작업을 쪼개었는데 이 과정에서 1024/64= 16 으로 나누어 떨어지는데 이 과정에서 코어수가 딱 한 소켓에 맞아떨어져서 반감 된것일수도 있는데, 이는 프로파일링이 더 필요해 보입니다.

저번 과제에서 계산한 2.15 TFlops 대비 13.9 퍼센트 정도의 성능을 내는데 이는 곱과 합을 번갈아 잘 배치 하지 못했고 캐쉬를 잘 활용하지 못해서 발생한것으로 보입니다.

#pragma omp parallel for schedule(static): 고정된 워크로드를 적절히 분배하는 방법입니다. 이러한 matmul처럼 고정된 워크로드가 오는 경우 유리해 보입니다.
#pragma omp parallel for schedule(dynamic, chunk_size)새로운 청크가 완료될때마다 새로운 작업을 부여받습니다.
#pragma omp parallel for schedule(guided, chunk_size) TCP와 비슷한 사상의 설계로 보입니다 처음에 작게 시작하여 점점 청크를 줄여서 최적점에 다가갑니다.
에서 chunk_size를 적절히 바꿔가면서 실험한 결과 결과적으로 유의미한 성능차이는 발생하지 않았습니다.

이 작업의 경우엔 매우 일정하게 비슷한 작업이 할당됩니다. 따라서 다이내믹하게 워크로드를 할당할 필요없이 모든작업이 비슷하게 끝나게 될 것입니다. 따라서 이러한 다이내믹한 스케줄링이 덜 필요할것입니다.