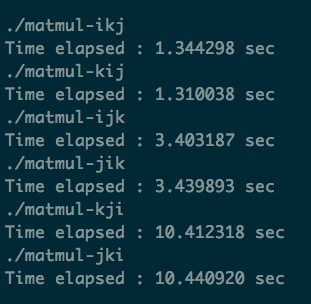
멀티코어 과제 4

2013-11392 김지현

# 문제 1번

## 파트 A

루프를 i, j, k 세 축으로 3차원 루프를 도는데, 이때 ijk, ikj, jki, jik, kij, jki 총 여섯가지 경우의 수가 있다. 각각의 모든 경우의 수를 실험해보면, 실행 결과가 주로 맨 마지막 루프를 어느 축으로 도느냐에 의해 거의 결정되고 앞의 두 축의 순서는 큰 영향이 없다는 사실을 알 수 있다.



위 실험결과는 1024\*1024 크기의 행렬의 곱에 대해 실험한것으로, j축 연산을 맨 마지막으로 수행하였을때 제일 빠른것을 알 수 있다.

## 파트 B

Tile 크기를 너무 작게하면, 불필요한 Loop 코드에 의한 오버헤드가 발생하고, Tile 크기를 너무 크게하면 eviction이 자주 일어나 코드가 cache-unfriendly하게 변한다. Tile의 크기를 변화시켜가며 실험시켜, 균형점을 알아낼 수 있다.

실험 결과 Tile 크기가 4~8인 경우 최적의 결과를 얻어내는것을 알 수 있다.

## 파트 C

스레드 갯수와 작업량 수를 조절해가며 수행성능을 비교해보면 Weak Scalability와 Strong Scalability가 각각 얼마인지 알 수 있다. 이때, 노드당 코어 숫자는 한정되어있고 코어 수를 넘는 스레드를 생성하여 실행시키는것은 큰 의미가 없으므로 주의하도록 한다. 천둥의 경우 계산 노드는 총 16코어로, 16스레드 이상은 실험시키지 않았다.

구스타프슨의 법칙에 대입하면, 병렬화되어 실행되는 코드의 비율이 78.8%로, 비교적 우수한 결과가 나왔다.

Strong Scaling Efficiency는 위와 같이, 스레드 숫자가 늘어나도 효율이 저해되지 않고 일정 비율을 유지하고, Weak Scaling Efficiency 또한 완만하게 감소하지만 작업 로드와 스레드가 커질수록 효율이 반으로 떨어지는것을 볼 수있다.

# 문제 2번

이 문제는 1번과는 다르게, 순차실행이 차지하는 부분의 비중이 커서 일정 이상으로는 병렬화가 쉽게 되지 못하는 모습을 보여줬다.

계산 노드는 16코어짜리 CPU로, 16개 이상의 스레드는 결과가 큰 의미가 없음에 유의하라.

1번 문제와는 다르게 Strong Scaling Efficiency가 스레드 숫자를 늘림에따라 급격하게 감소하는것을 볼 수 있다. Weak Scaling Efficiency 상대적으로 Strong Scaling Efficiency 그래프보다 경사가 원만하여, K-means 문제는 큰 계산을 많은 스레드가 나눠서 하였을때 보다 효율적임을 알 수 있었다.