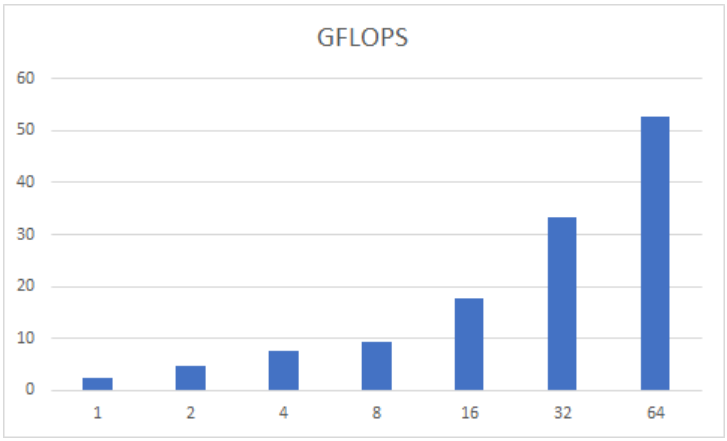
HW 3 (2021-27764 안지수)

1. Matrix multiplication with OpenMP

(a) Matrix multiplication 방법 중 cache friendly하게 연산을 수행하는 알고리즘을 변형한 방식으로 병렬화를 진행했다. (cache friendly한 방식은 matrix를 cache에 한번에 load할 수 있게 작은 block으로 나누어 계산하는 방식이다) 처음에는 전체 for문을 병렬화하는 방식으로 진행하다 실험을 통해 가장 성능이 잘 나오는 순서로 연산을 수행하도록 변형을 진행했다.

(b)

```
./main -v -t 1 4096 4096 4096 > log.txt
./main -v -t 2 4096 4096 4096 >> log.txt
./main -v -t 4 4096 4096 4096 >> log.txt
./main -v -t 8 4096 4096 4096 >> log.txt
./main -v -t 16 4096 4096 4096 >> log.txt
./main -v -t 32 4096 4096 4096 >> log.txt
./main -v -t 64 4096 4096 4096 >> log.txt
```



다음과 같은 방식으로 연산을 수행하여 결과를 얻을 수 있었다. 예상처럼 쓰래드가 늘어날수록 성능은 증가 하지만 성능이 linear하게 증가하지는 않음을 확인할 수 있었다. 이는 병렬화 할 수 있는 부분이 한정되어 있고, 병렬화를 하더라도 memory I/O에서 발생하는 dependance 때문이라고 생각할 수 있었다.

(c)

```
Initializing... done!
Calculating...(iter=0) 0.019744 sec
Validating...
Result: VALID
Reference time: 0.002311 sec
Reference throughput: 75.561826 GFLOPS
Your Avg. time: 0.019744 sec
Your Avg. throughput: 8.845292 GFLOPS
```

(d)

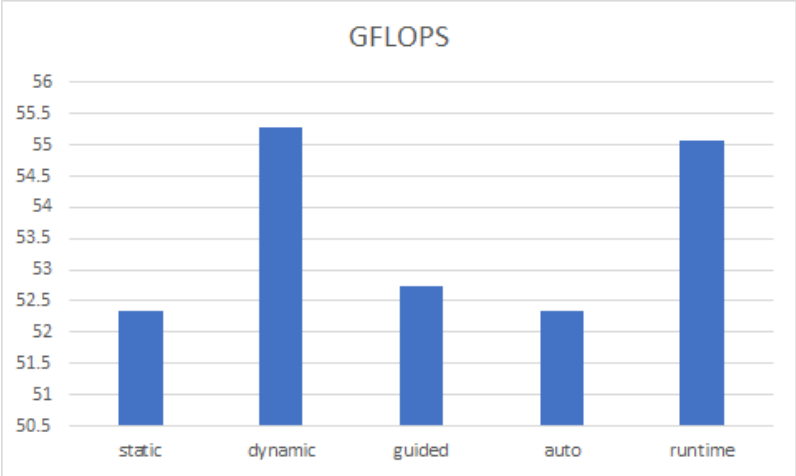
```
Initializing... done!
Warming up... 4.081155 sec
Warming up... 3.872720 sec
Warming up... 3.896632 sec
Calculating...(iter=0) 3.856984 sec
Calculating...(iter=1) 3.948091 sec
Calculating...(iter=2) 3.840151 sec
Validating...
Result: VALID
Reference time: 1.998218 sec
Reference throughput: 68.780748 GFLOPS
Your Avg. time: 3.881742 sec
Your Avg. throughput: 35.406515 GFLOPS
shpc121@a00:~/snu_shpc21/hw3$
```

OpenMP schedule Clause

(a) static, dynamic, guided auto, runtime

(b)

```
./main -v -t 64 4096 4096 4096
```



64개의 thread로 병렬 프로그래밍을 수행했을 때 다음과 같이 dynamic과 runtime에서 비교적 좋은 성능을 내고, static, guided, auto에서 비교적 좋지 않은 성능을 내는 것을 확인할 수 있었다.