멀티코어 HW3 보고서

2016-11464 정재훈

1. 서론

본 보고서에서는 OpenCL을 사용하여 matrix multiplication을 병렬화를 이용하여 수행하고 그 방법과 matrix multiplication을 실행해본 결과와 결과에 대한 분석을 하였다.

2. 병렬화 방법

group id와 local id를 사용하였다. 각 그룹에서 local id들은 서로 병렬화 되어 진행되고, 이들을 synchronize 하기 위하여 모든 병렬 처리가 끝나기를 기다려야 하는 곳에서는 barrier를 사용하였다. 그룹은 matrix를 16-by-16 단위로 tiling 하기 위하여 사용했는데, local id들을 가진 계산 유닛들은 한 group씩 처리를 끝마칠 것이고(한 group의 처리가 끝나면 결과 matrix의 16-by-16 영역 하나가 채워진다), 결국 matrix의 곱을 계산해낼 것이다.


또한 float4 data type를 사용하여 4개의 데이터를 한 번에 계산하여 곱하고 더하는 instruction이 한 개씩 계산할 때보다 조금 더 빠르도록 하였다.
커널 코드는 kernel.cl에 작성한 후 fread를 통해서 읽어서 string으로 바꿔서 사용하였다.

3. 실행 결과

```
Problem size:   3072 x 3072 x 3072

 Initializing ...       done!
 Calculating ...        done!
 Validation off.

Elapsed time:   0.134 sec (133.519 ms)
```

```
Problem size:   4096 x 4096 x 4096

 Initializing ...       done!
 Calculating ...        done!
 Validation off.

Elapsed time:   0.260 sec (259.797 ms)
```

```
Problem size:   6144 x 6144 x 6144

 Initializing ...       done!
 Calculating ...        done!
 Validation off.

Elapsed time:   0.769 sec (768.810 ms)
```

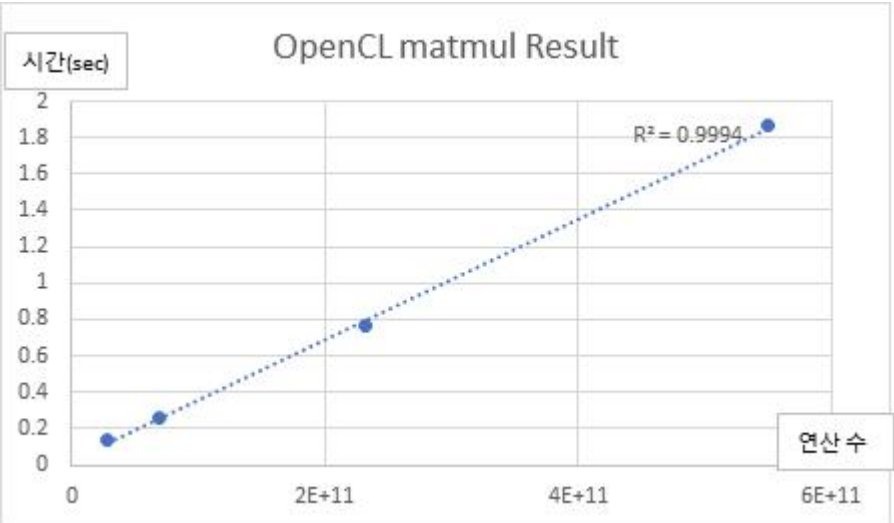
```
Problem size:   8192 x 8192 x 8192

 Initializing ...       done!
 Calculating ...        done!
 Validation off.

Elapsed time:   1.866 sec (1865.870 ms)
```

차례로 3072, 4096, 6144, 8192 개의 행과 열을 가진 두 matrix의 곱 연산에 걸린 시간이다. validation은 3072 실행 결과 성공하며(스크린샷에는 없지만 개인적으로 확인함), 그 이상의 경우 시간이 매우 오래 걸려서 확인해보지는 않았지만, tiling 원리를 생각할 때 16 이상의 16의 배수에 대하여 모두 정확한 결과를 낼 것으로 추정된다.

4. 결과 분석




가로축을 연산 횟수(N^3), 세로축을 걸린 시간(초) 로 그래프를 그려 본 결과 거의 일직선을 이루고 있는 것으로 보아 네 실행에서 거의 같은 수의 계산 유닛들이 계산에 참여한 것으로 판단할 수 있다. (아마도 256개 일 것)

5. 토의

group size를 32*32 이상으로 하면 정확한 결과가 나오지 않은 것으로 보아 1024개 이상의 계산 유닛은 천둥에 존재하지 않는 것 같다.

6. 특이사항

없다.