하드웨어시스템설계 LAB02

2016-19516 손상준 / 2016-10454 이지원

0. 행렬과 벡터의 곱셈에 대하여

주어진 large mat 과 input 에서, 가장 기본적인 행렬과 벡터의 곱셈은 다음과 같이 나타난다.

$$\text{large\_mat} : m * n \text{ matrix}, \text{input} : n * r \text{ matrix}$$

$$Result_{ij} = \sum_{k=1}^{n} large\_mat_{ik} * input_{kj}$$

위의 방법으로 실행시켰을 때,

결과는 다음과 같다.

```
root@e2ef1cddef7b:~/21s-hardware-system-design/lab02# python eval.py
read dataset...
('images', (10000, 28, 28))
create network...
run test...
{'accuracy': 0.9159,
 'avg_num_call': 0,
 'm_size': 64,
 'total_image': 10000,
 'total_time': 25.758726119995117,
 'v_size': 64}
```

1. Tiling 방법

행렬과 벡터의 곱셈에서, 행렬과 벡터를 작은 크기로 나누어 곱 연산을 수행하고, 결과값을 누적하여 최종 값을 구한다. 다음은 블록 크기 (M, V) : (64, 64), (16, 16), (8, 16), (16, 8) 에 대하여 수행한 결과이다.

```
root@e2ef1cddef7b:~/21s-hardware-system-design/lab02# python eval.py
read dataset...
('images', (10000, 28, 28))
create network...
run test...
{'accuracy': 0.9159,
 'avg_num_call': 627,
 'm_size': 64,
 'total_image': 10000,
 'total_time': 33.638646841049194,
 'v_size': 64}
```

2016-19516 손상준
2016-10454 이지원

```
root@e2ef1cddef7b:~/21s-hardware-system-design/lab02# python eval.py 16 16
read dataset...
('images', (10000, 28, 28))
create network...
run test...
{'accuracy': 0.9159,
 'avg_num_call': 9375,
 'm_size': 16,
 'total_image': 10000,
 'total_time': 36.78242588043213,
 'v_size': 16}
root@e2ef1cddef7b:~/21s-hardware-system-design/lab02# python eval.py 16 8
read dataset...
('images', (10000, 28, 28))
create network...
run test...
{'accuracy': 0.9159,
 'avg_num_call': 18750,
 'm_size': 16,
 'total_image': 10000,
 'total_time': 53.67651700973511,
 'v_size': 8}
```

2016-19516 손상준
2016-10454 이지원

```
root@e2ef1cddef7b:~/21s-hardware-system-design/lab02# python eval.py 8 16
read dataset...
('images', (10000, 28, 28))
create network...
run test...
{'accuracy': 0.9159,
 'avg_num_call': 18750,
 'm_size': 8,
 'total_image': 10000,
 'total_time': 47.869489908218384,
 'v_size': 16}
```

2016-19516 손상준
2016-10454 이지원

주의할 점은, tile의 행과 열의 크기가 원래 행렬의 행과 열에 대하여 나누어 떨어지지 않으면, tile의 크기를 조절하여 곱 연산을 수행해 주어야 한다.

2. 결론

결과를 관찰하였을 때, 동일한 training set이 사용되어 정확도는 모두 같고, M과 V의 크기가 클수록 전체 시간이 줄어드는 것을 확인할 수 있었다.

비록 이번 랩에서는 28*28의 비교적 크기가 작은 행렬을 이용하였지만, FPGA보드가 64 * 64 크기의 행렬과 벡터의 곱셈을 지원하기 때문에, 크기가 큰 행렬, 벡터의 곱셈을 계산해야 하는 경우 위의 방법이 필수적일 것이라는 생각이 들었다.

또한, 이러한 방법이 더 빠르게 행렬과 벡터의 곱셈 결과를 얻는 데도 사용될 수 있다고 생각하였다. Tiling 방법에서 각 tile들의 곱셈은 서로 독립적이기 때문에, multi-thread를 이용하여 계산하게 되면 더 빠르게 결과값을 도출할 수 있다고 생각한다.